TUTORIAL 4

Graphic Design with CSS

Creating a Graphic Design for a Genealogy Website

OBJECTIVES

Session 4.1
- Create a figure box
- Add a background image
- Add a border to an element
- Create rounded borders
- Create a graphic border

Session 4.2
- Create a text shadow
- Create a box shadow
- Create linear and radial gradients
- Set the opacity of an element

Session 4.3
- Apply a 2D and 3D transformation
- Apply a CSS filter
- Create an image map

Case | *Tree and Book*

Kevin Whitmore is the founder of *Tree and Book*, a social networking website for people interested in documenting their family histories, creating online photo albums, and posting stories and information about members of their extended families. He has come to you for help in upgrading the site's design. Kevin wants to take advantage of some of the CSS styles that can be used to add interesting visual effects to his site in order to give his website more impact and visual interest.

STARTING DATA FILES

tutorial
tb_genta_txt.html
tb_komatsu_txt.html
tb_visual1_txt.css
tb_visual2_txt.css
+ 21 files

review
tb_ferris_txt.html
tb_kathleen_txt.html
tb_visual3_txt.css
tb_visual4_txt.css
+ 16 files

code1
code4-1_txt.html
code4-1_back_txt.css
+ 2 files

code2
code4-2_txt.html
code4-2_grad_txt.css
+ 2 files

code3
code4-3_txt.html
code4-3_cube_txt.css
+ 6 files

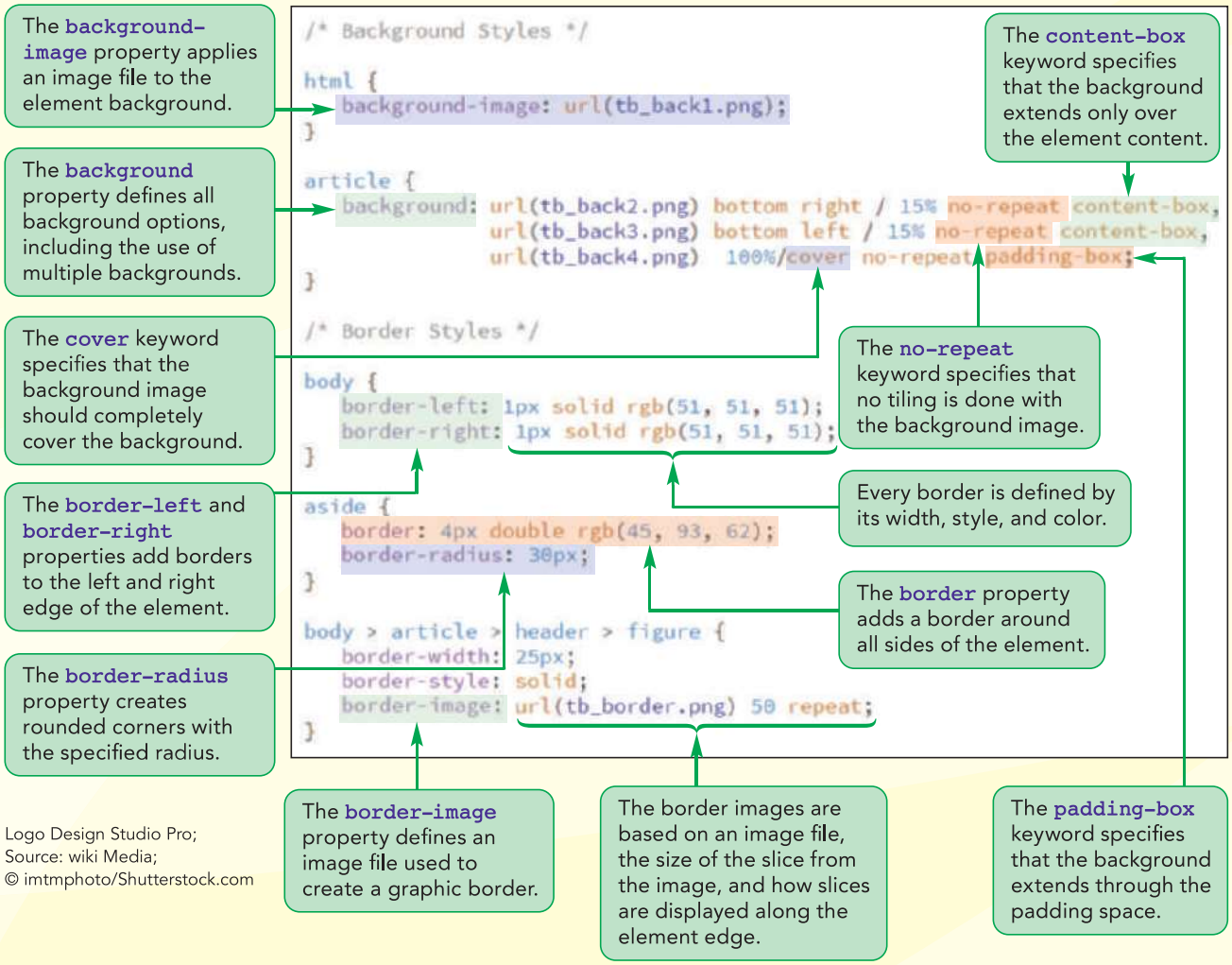
code4
code4-4_txt.html
debug4-4_txt.css
+ 2 files

case1
sf_torte_txt.html
sf_effects_txt.css
+ 11 files

case2
cf_home_txt.html
cf_effects_txt.css
+ 9 files

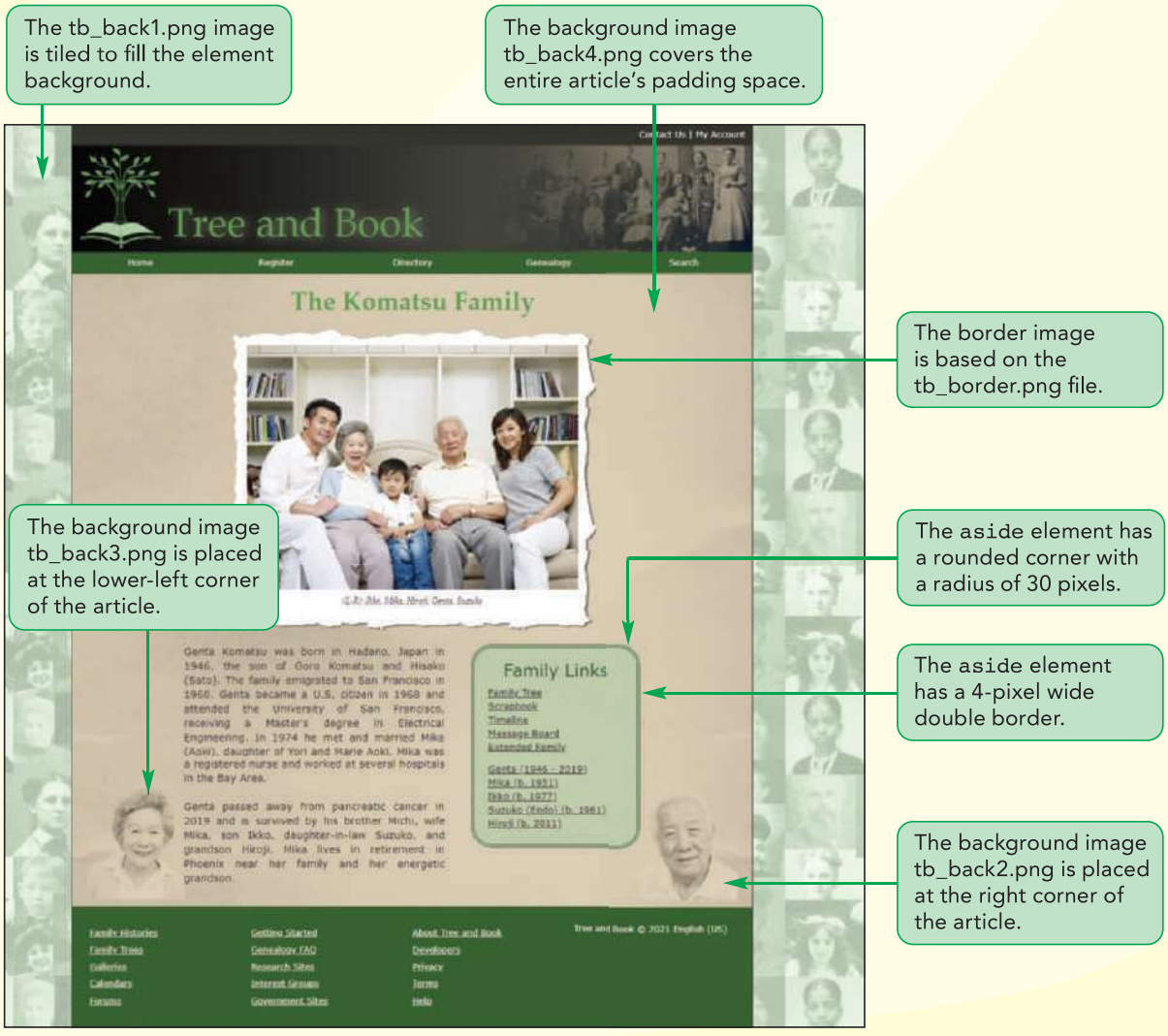
demo
24 demo pages
+ 21 files

Session 4.1 Visual Overview:



Logo Design Studio Pro;
Source: wiki Media;
© imtmphoto/Shutterstock.com

Backgrounds and Borders



Source: Wikimedia Commons; Design Studio Pro; imtmphoto/Shutterstock.com; imtmphoto/Shutterstock.com

Creating Figure Boxes

So far your work with CSS visual design styles has been limited to typographical styles and styles that modify the page's color scheme. In this tutorial, you'll explore other CSS styles that allow you to add figure boxes, background textures, background images, and three-dimensional effects to your web pages.

You'll start by examining how to work with figure boxes. In books and magazines, figures and figure captions are often placed within a separate box that stands apart from the main content of the article, using the following `figure` and `figcaption` elements:

```
<figure>
   content
   <figcaption>caption text</figcaption>
</figure>
```

where *content* is the content that will appear within the figure box and *caption text* is the description text that accompanies the figure. The `figcaption` element is optional and can be placed either directly before or directly after the figure box content. For example, the following code marks a figure box containing the tb_komatsu.png image file with the caption *(L-R): Ikko, Mika, Hiroji, Genta, Suzuko*.

```
<figure>
   <img src="tb_komatsu.png" alt="family portrait" />
   <figcaption>(L-R): Ikko, Mika, Hiroji, Genta, Suzuko</figcaption>
</figure>
```

TIP

The semantic difference between the `figure` and `aside` elements is that the `figure` element should be used for content that is directly referenced from within an article while the `aside` element is used for extraneous content.

While the `figure` element is used to contain an image file, it can also be used to mark any page content that you want to stand apart from the main content of an article. For instance, the `figure` element could contain a text excerpt, as the following code demonstrates:

```
<figure>
  <p>'Twas brillig, and the slithy toves<br />
      Did gyre and gimble in the wabe;<br />
      All mimsy were the borogoves,<br />
      And the mome raths outgrabe.</p>
  <figcaption>
    <cite>Jabberwocky, Lewis Carroll, 1832-98</cite>
  </figcaption>
</figure>
```

Kevin plans on using figure boxes throughout the Tree and Book website to mark up family and individual photos along with descriptive captions. He's created a set of sample pages for the Komatsu family that you will work on to learn about HTML and CSS visual elements and styles. Open the family's home page and create a figure box displaying the family portrait along with a descriptive caption.

To create a figure box:

1. Use your editor to open the **tb_komatsu_txt.html** file from the html04 ▸ tutorial folder. Enter ***your name*** and ***the date*** in the comment section of the file and save it as **tb_komatsu.html**.

   For this web page, you'll work with a new style sheet named tb_visual1.css. Kevin has already created a reset style sheet and a typographical style sheet in the tb_reset.css and tb_styles1.css files respectively.

2. Within the document head, insert the following `link` elements to link the page to the tb_reset.css, tb_styles1.css, and tb_visual1.css style sheet files.

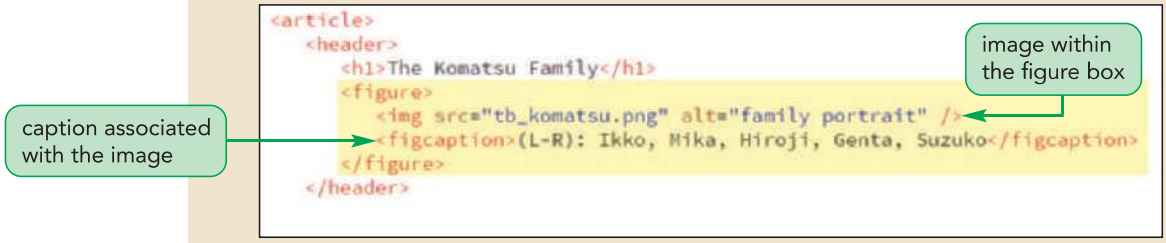
```
<link href="tb_reset.css" rel="stylesheet" />
<link href="tb_styles1.css" rel="stylesheet" />
<link href="tb_visual1.css" rel="stylesheet" />
```

3. Scroll down to the `article` element and, directly after the `h1` element, insert the following code for the figure box displaying the Komatsu family portrait.

```
<figure>
   <img src="tb_komatsu.png" alt="family portrait" />
   <figcaption>(L-R): Ikko, Mika, Hiroji,
               Genta, Suzuko
   </figcaption>
</figure>
```

Figure 4–1 highlights the code for the family portrait figure box.

Figure 4–1 Inserting a figure box



4. Take some time to review the content and structure of the rest of the document and then save your changes to the file.

Format the appearance of the figure box by adding new style rules to the tb_visual1.css style sheet file.

To format and view the figure box:

1. Use your editor to open the **tb_visual1_txt.css** files from the html04 ▸ tutorial folder. Enter ***your name*** and ***the date*** in the comment section of the file and save it as **tb_visual1.css**.
2. Scroll down to the Figure Box Styles section at the bottom the document and insert the following style rule for the `figure` element:

```
figure {
   margin: 20px auto 0px;
   width: 80%;
}
```

3. Add the following style to format the appearance of the image within the figure box:

```
figure img {
   display: block;
   width: 100%;
}
```

4. Finally, insert the following rule for the figure caption:

```
figure figcaption {
   background-color: white;
   font-family: 'Palatino Linotype', Palatino,
                'Times New Roman', serif;
   font-style: italic;
   padding: 10px 0;
   text-align: center;
}
```

Figure 4–2 highlights the style rules for the figure box, image, and caption.

Figure 4–2 Formatting the figure box and caption

figure box is 80% of the width of the header and centered horizontally

figure image is displayed as a block with a width equal to the figure box

figure caption is centered and displayed in a serif italic font on a white background

```
/* Figure Box Styles */

figure {
   margin: 20px auto 0px;
   width: 80%;
}

figure img {
   display: block;
   width: 100%;
}

figure figcaption {
   background-color: white;
   font-family: 'Palatino Linotype', Palatino, 'Times New Roman', serif;
   font-style: italic;
   padding: 10px 0;
   text-align: center;
}
```

5. Save your changes to the file and then open the **tb_komatsu.html** file in your browser. Figure 4–3 shows the initial appearance of the page.

Figure 4–3 Initial design of the Komatsu family page

Source: Design Studio Pro; Source: Wikimedia Commons; © imtmphoto/Shutterstock.com

With all of the content for the Komatsu Family page now added, you will start working on enhancing the page's appearance, starting with the CSS background styles.

INSIGHT

Choosing your Graphic File Format

Graphic files on the web fall into two basic categories: vector images and bitmap images. A **vector image** is an image comprised of lines and curves that are based on mathematical functions. The great advantage of vector images is that they can be easily resized without losing their clarity and vector files tend to be compact in size. The most common vector format for the web is **SVG (Scalable Vector Graphics)**, which is an XML markup language that can be created using a basic text editor and knowledge of the SVG language.

A **bitmap image** is an image that is comprised of pixels in which every pixel is marked with a different color. Because a graphic file can be comprised of thousands of pixels, the file size of a bitmap image is considerably larger than the file size of a vector image. The most common bitmap formats on the web are GIF, JPEG, and PNG.

GIF (Graphic Interchange Format) is the oldest standard with a palette limited to 256 colors. GIF files, which tend to be large, have two advantages: first, GIFs support transparent colors and second, GIFs can be used to create animated images. Because GIFs have a limited color palette, they are unsuitable for photos. The most popular photo format is **JPEG (Joint Photographic Experts Group)**, which supports a palette of over 16 million colors. JPEGs also support file compression, allowing a bitmap image to be stored at a smaller file size than would be possible with other bitmap formats. JPEGs do not support transparent colors or animations.

The **PNG (Portable Network Graphics)** format was designed to replace GIFs with its support for several levels of transparent colors and palette of millions of colors. A PNG file can also be compressed, creating a file that is considerably smaller and, therefore, takes up considerably less space than its equivalent GIF file. PNG files also contain color correction information so that PNGs can be accurately rendered across a variety of display devices.

In choosing a graphic format for your website, the most important consideration is often file size; you want to choose the smallest size that still gives you an acceptable image. This combination means that users will view a quality image but they will not have to wait for the graphic file to download. In addition to file size, you want to choose a format that supports a large color palette. For these reasons, most graphics on the web are now in either JPEG or PNG format, though GIFs are still often found on legacy sites.

Exploring Background Styles

Thus far, your design choices for backgrounds have been limited to color using either the RGB or HSL color models. CSS also supports the use of images for backgrounds through the following `background-image` style:

```
background-image: url(url);
```

where *url* specifies the name and location of the background image. For example, the following style rule uses the trees.png file as the background of the page body.

```
body {
   background-image: url(trees.png);
}
```

This code assumes that the trees.png file is in the same folder as the style sheet; if the figure is not in the same folder, then you will have to include path information pointing to the folder location in which the image file resides.

Tiling a Background Image

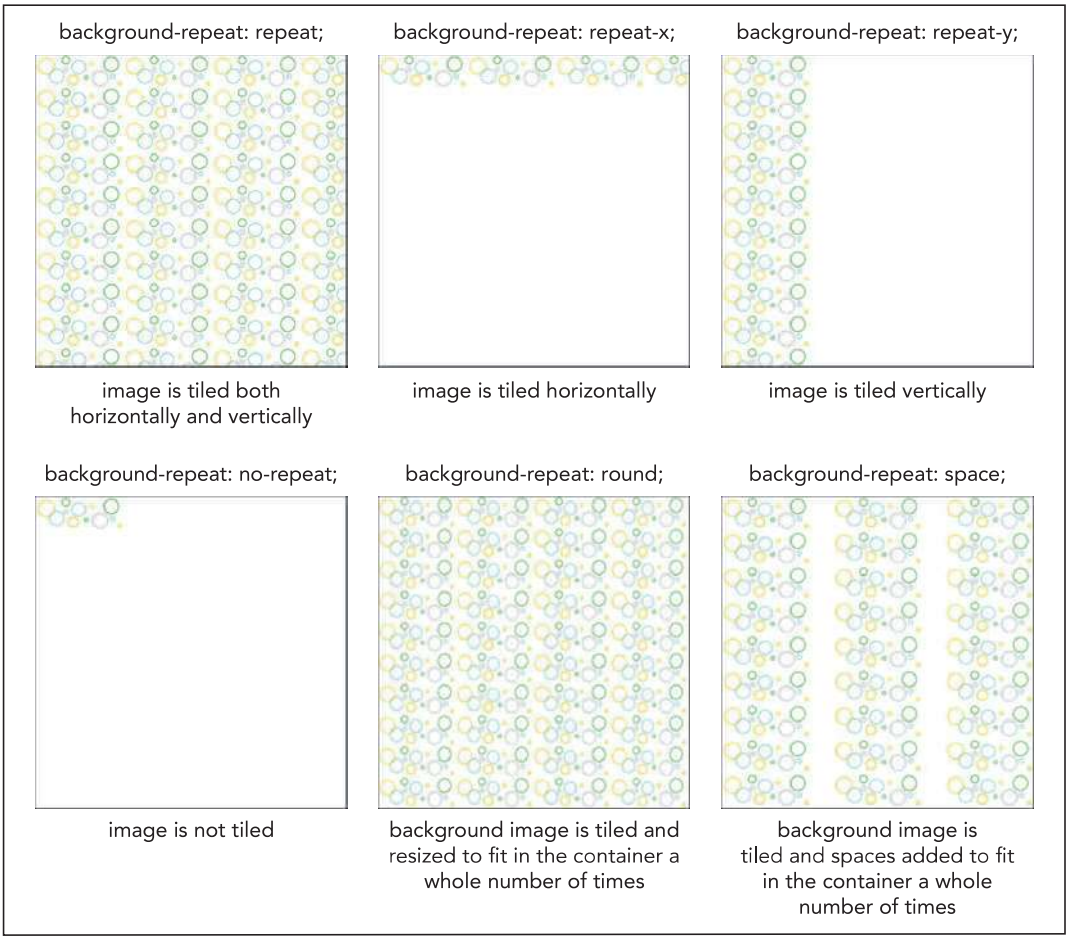
The default browser behavior is to place the background image at the top-left corner of the element and repeat the image in both the vertical and horizontal direction until the background is filled. This process is known as **tiling** because of its similarity to the process of filling up a floor or other surface with tiles.

You can specify the type of tiling to be applied to the background image, or even turn off tiling, by applying the following `background-repeat` style:

```
background-repeat: type;
```

where *type* is `repeat` (the default), `repeat-x`, `repeat-y`, `no-repeat`, `round`, or `space`. Figure 4–4 displays the effect of each `background-repeat` type.

Figure 4–4 Examples of background-repeat types



REFERENCE

Adding a Background Image

- To add an image to the background, use the CSS style

  `background-image: url(url);`

  where *url* specifies the name and location of the background image.
- To specify how the image should be tiled, use

  `background-repeat: type;`

  where *type* is `repeat` (the default), `repeat-x`, `repeat-y`, `no-repeat`, `round`, or `space`.

Kevin has supplied you with an image file, tb_back1.png to fill the background of the browser window. Use the default option for tiling so that the image is displayed starting from the top-left corner of the window and repeating until the entire window is filled.

To add a background image to the browser window:

1. Return to the **tb_visual1.css** file in your editor.
2. Go to the HTML Styles section and add the following style rule to change the background of the browser window:

```
html {
   background-image: url(tb_back1.png);
}
```

   Note that because you are using the default setting for tiling the background image, you do not need to include the `background-repeat` style rule. Figure 4–5 highlights the new style rule.

Figure 4–5 Defining a background image

tiles the tb_back1.png image file across the browser window background

```
/* HTML Styles */

html {
   background-image: url(tb_back1.png);
}
```

3. Save your changes to the file and then reload tb_komatsu.html in your browser. Figure 4–6 shows the tiled background in the browser window.

Figure 4–6 Tiled background image in the browser window

Source: Design Studio Pro; Source: Wikimedia Commons; © imtmphoto/Shutterstock.com

Note that the page body covers part of the tiled images in the browser window. However, even though the background images are hidden, the tiling still continues behind the page body.

Attaching the Background Image

A background image is attached to its element so that as you scroll through the element content, the background image scrolls with it. You can change the attachment using the following `background-attachment` property

```
background-attachment: type;
```

where *type* is `scroll` (the default), `fixed`, or `local`. The `scroll` type sets the background to scroll with the element content. The `fixed` type creates a background that stays in place even as the element content is scrolled horizontally or vertically. Fixed backgrounds are sometimes used to create **watermarks**, which are translucent graphics displayed behind the content with a message that the content material is copyrighted or in draft form or some other message directed to the reader. The `local` type is similar to `scroll` except that it is used for elements, such as scroll boxes, to allow the element background to scroll along with the content within the box.

Setting the Background Image Position

By default, browsers place the background image in the element's top-left corner. You can place the background image at a different position using the following `background-position` property:

```
background-position: horizontal vertical;
```

TIP

Background coordinates are measured from the top-left corner of the background to the top-left corner of the image.

where *horizontal* and *vertical* provide the coordinates of the image within the element background expressed using one of the CSS units of measure or as a percentage of the element's width and height. For example, the following style places the image 10% of the width of the element from the left edge of the background and 20% of the element's height from the background's top edge.

```
background-position: 10% 20%;
```

If you specify a single value, the browser applies that value to both the horizontal and vertical position. Thus, the following style places the background image 30 pixels from the element's left edge and 30 pixels down from the top edge.

```
background-position: 30px;
```

You can also place the background image using the keywords `left`, `center`, and `right` for the horizontal position and `top`, `center`, and `bottom` for the vertical position. The following style places the background image in the bottom-right corner of the element.

```
background-position: right bottom;
```

Typically, the `background-position` property is only useful for non-tiled images because, if the image is tiled, the tiled image fills the background and it usually doesn't matter where the tiling starts.

Defining the Extent of the Background

You learned in Tutorial 2 that every block element follows the Box Model in which the element content is surrounded by a padding space and beyond that a border space (see Figure 2-38). However, the element's background is defined, by default, to extend only through the padding space and not to include the border space. You can change this definition using the following `background-clip` property:

```
background-clip: type;
```

TRY IT

You can explore the impact of different CSS background styles using the demo_background.html file in the html04 ▸ demo folder.

where *type* is `content-box` (to extend the background only through the element content), `padding-box` (to extend the background through the padding space), or `border-box` (to extend the background through the border space). For example, the following style rule defines the background for the page body to extend only as far as the page content. The padding and border spaces would not be considered part of the background and thus would not show any background image.

```
body {
   background-clip: content-box;
}
```

Because the background extends through the padding space by default, all coordinates for the background image position are measured from the top-left corner of that padding space. You can choose a different context by applying the following `background-origin` property:

```
background-origin: type;
```

where *type* is once again `content-box`, `padding-box`, or `border-box`. Thus, the following style rule places the background image at the bottom-left corner of the page body content and not the bottom-left corner of the padding space (which would be the default).

```
body {
   background-position: left bottom;
   background-origin: content-box;
}
```

Based on this style rule, the padding space of page body would not have any background image or color, other than what would be defined for the browser window itself.

Sizing and Clipping an Image

The size of the background image is equal to the size stored in the image file. To specify a different size, apply the following `background-size` property:

```
background-size: width height;
```

where *width* and *height* are the width and height of the image in one of the CSS units of length or as a percentage of the element's width and height. The following style sets the size of the background image to 300 pixels wide by 200 pixels high.

```
background-size: 300px 200px;
```

CSS also supports the sizing keywords `auto`, `cover`, and `contain`. The `auto` keyword tells the browser to automatically set the width or height value based on the dimensions of the original image. The following style sets the height of the image to 200 pixels and automatically scales the width to keep the original proportions of the image:

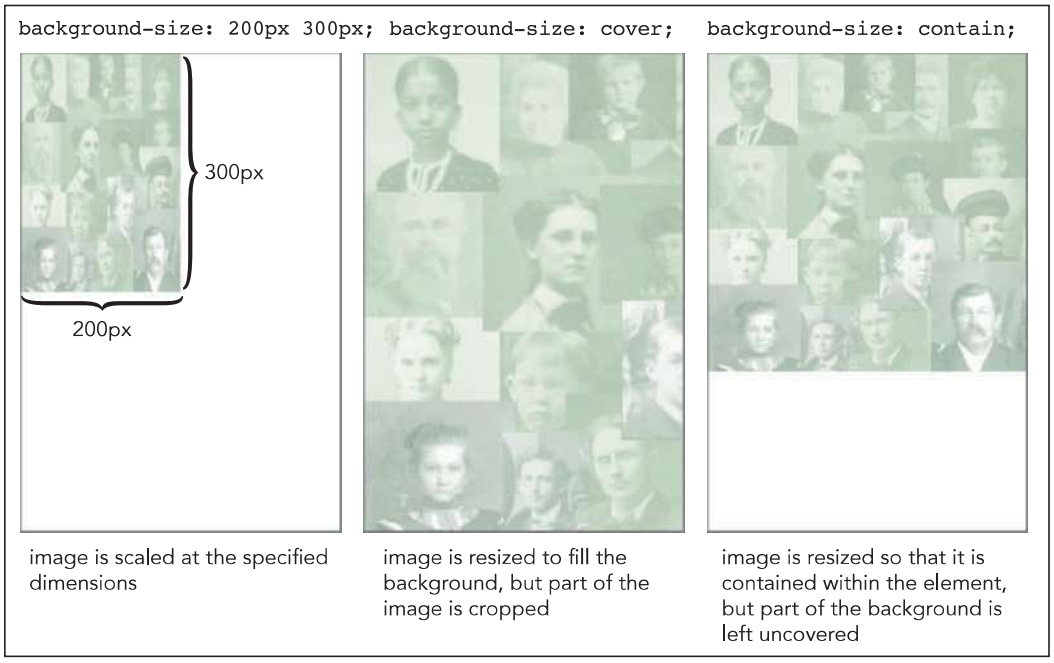
```
background-size: auto 200px;
```

TIP

If you specify only one size value, the browser applies it to the image width and scales the height proportionally.

The `cover` keyword tells the browser to resize the image to cover all of the element background while still retaining the image proportions. Depending on the size of the element, this could result in some of the background image being cropped. The `contain` keyword scales the image so that it's completely contained within the element, even if that means that not all of the element background is covered. Figure 4–7 displays examples of a background set to a specific size, as well as resized to either cover the background or to have the image completely contained within the background.

Figure 4–7 Examples of background-size types



Source: Wikimedia Commons

REFERENCE

Setting Background Image Options

- To specify how the image is attached to the background, use

  `background-attachment: type;`

  where *type* is `scroll` (the default), `fixed`, or `local`.
- To set the position of the background image, use

  `background-position: horizontal vertical;`

  where *horizontal* and *vertical* provide the coordinates of the image within the element background.
- To define the extent of the background, use

  `background-clip: type;`

  where *type* is `content-box`, `padding-box` (the default), or `border-box`.
- To define how position coordinates are measured, use

  `background-origin: type;`

  where *type* is `content-box`, `padding-box` (the default), or `border-box`.

The background Property

All of these different background options can be organized in the following `background` property:

```
background: color url(url) position / size repeat attachment
origin clip;
```

TRY IT

You can explore the CSS background style using the demo_background2.html file in the html04 ▸ demo folder.

where *color* is the background color, *url* is the source of the background image, *position* is the image's position, *size* sets the image size, *repeat* sets the tiling of the image, *attachment* specifies whether the image scrolls with the content or is fixed, *origin* defines how positions are measured on the background, and *clip* specifies the extent over which the background is spread. For example, the following style rule sets the background color to ivory and then uses the draft.png file as the background image fixed at the horizontal and vertical center of the page body and sized at 10% of the body's width and height:

```
body {
   background: ivory url(draft.png)
               center center / 10% 10%
               no-repeat fixed content-box content-box;
}
```

The rest of the property sets the image not to repeat and to use the content box for defining the background origin and clipping. Note that the page body will have an ivory background color at any location where the draft.png image is not displayed. If you don't specify all of the option values, the browser will assume the default values for the missing options. Thus, the following style rule places the draft.png at the horizontal and vertical center of the page body without tiling:

```
body {
   background: ivory url(draft.png) center center no-repeat;
}
```

TIP

The background property includes the "/" character only when you need to separate the image position value from the image size value.

Since no *size*, *attachment*, *origin*, and *clip* values are specified, the size of the image will be based on the dimensions from the image file, the image will scroll with the body content, and the background origin and clipping will extend through the page body's padding space.

Kevin wants you to include a semi-transparent image of the family patriarch, Genta Komatsu, as a background image placed in the lower-right corner of the article on the Komatsu family. Add a style rule to the tb_visual1.css file to display the tb_back2.png image within that element without tiling.

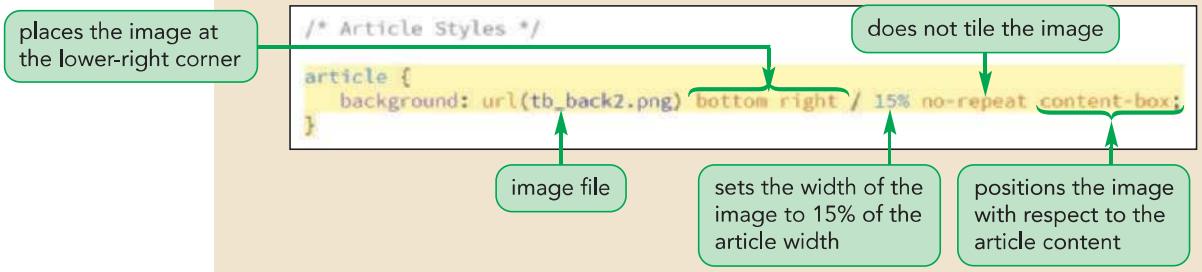
To add a background image to the page article:

1. Return to the **tb_visual1.css** file in your editor and scroll down to the Article Styles section.
2. Add the following style rule:

```
article {
    background: url(tb_back2.png) bottom right / 15%
                no-repeat content-box;
}
```

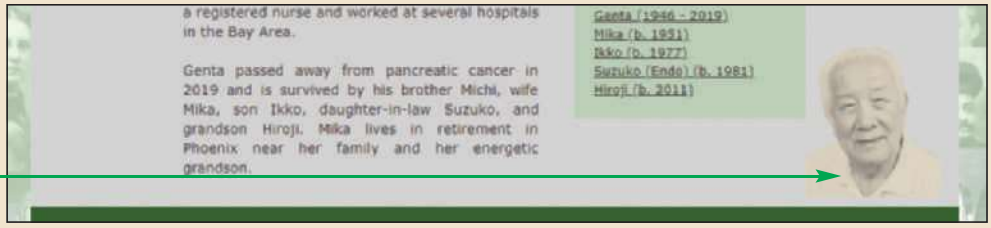
Figure 4–8 highlights the style rule applied to the page article.

Figure 4–8 Adding a background to the page article



3. Save your changes and then reload tb_komatsu.html in your browser. Figure 4–9 shows the placement of the background image.

Figure 4–9 Placement of the background image



Source: Wikimedia Commons; © imtmphoto/Shutterstock.com

Kevin likes the addition of the image of Genta Komatsu and would like you to add another background image showing the family matriarch, Mika Komatsu, and a third image giving the article a paper-textured background.

Adding Multiple Backgrounds

To add multiple backgrounds to the same element, you list the backgrounds in the following comma-separated list:

```
background: background1, background2, …;
```

where *background1*, *background2*, and so on are the properties for each background. For example the following style rule applies three different backgrounds to the `header` element:

```
header {
   background: url(back2.png) top left no-repeat,
               url(back1.png) bottom right no-repeat,
               rgb(191, 191, 191);
}
```

TIP

Always list the background color last so that it provides the foundation for your background images.

Backgrounds are added in the reverse order in which they're listed in the style rule. In this style rule, the background color is applied first, the back1.png background image is placed on top of that, and finally the back2.png background image is placed on top of those two backgrounds.

Individual background properties can also contain multiple options placed in a comma-separated list. The following style rule creates the same multiple backgrounds for the `header` element without using the `background` property:

```
header {
   background-image: url(back2.png), url(back1.png);
   background-position: top left, bottom right;
   background-repeat: no-repeat;
   background-color: rgb(191, 191, 191);
}
```

Note that if a background style is listed once, it is applied across all of the backgrounds. Thus the `background-color` and the `background-repeat` properties are used in all the backgrounds.

Revise the style rule for the `article` element to add two more backgrounds.

To add a background image to the page article:

1. Return to the **tb_visual1.css** file in your editor and return to the Article Styles section.
2. Type a comma after the first background listed for the `article` element and before the semicolon (;), then press **Enter**.

The properties for multiple backgrounds need to be separated by commas.

3. Be sure the insertion point is before the semicolon (;), then add the following code to display two more background images followed by a background color:

```
url(tb_back3.png) bottom left / 15% no-repeat content-box,
url(tb_back4.png) 100%/cover no-repeat,
rgb(211, 211, 211)
```

The background color acts as a fallback design element and will not be displayed except for browsers that are incapable of displaying background images. Figure 4–10 displays the code for the multiple backgrounds applied to the page article.

Figure 4–10 Adding multiple background images

places the second background image at the lower-left corner of the article content with no tiling and a width of 15%

places the third background image, scaled to cover all of the padding box of the article without repeating

commas used to separate one background from the next

uses a gray color as the background if the browser doesn't support background images

```
/* Article Styles */

article {
   background: url(tb_back2.png) bottom right / 15% no-repeat content-box,
               url(tb_back3.png) bottom left / 15% no-repeat content-box,
               url(tb_back4.png) 100%/cover no-repeat,
               rgb(211, 211, 211);
}
```

Trouble? Be sure your code matches the code in Figure 4–10, including the commas used to separate the components in the list and the ending semicolon.

4. Save your changes and then reload tb_komatsu.html in your browser. Figure 4–11 shows the three background images displayed with the article.

Figure 4–11 Revised background for the page article

Source: Wikimedia Commons; © imtmphoto/Shutterstock.com

Kevin is pleased with the revised backgrounds for the browser window and the page article. Next, you will explore how to work with CSS border properties.

INSIGHT

Blending Backgrounds

Multiple backgrounds are stacked on top of each other, with the first background listed in the code placed on top of subsequent backgrounds. By default, the backgrounds on the top of the stack will obscure the lower-ordered backgrounds unless there is gap or a transparent color that allows the backgrounds at the bottom of the stack to appear. Thus, each background acts as its own background layer, separate in appearance from other backgrounds.

To combine multiple backgrounds into a single background, use the following `background-blend-mode` style:

```
background-blend-mode: type;
```

where *type* defines the method by which the backgrounds are combined. Possible *type* values include:

- `normal` backgrounds are stacked (the default)
- `multiply` background colors are multiplied, leading to a darker image combined of several colors
- `overlay` background colors are mixed to reflect the lightness and darkness of the colors
- `darken` backgrounds are replaced with the darkest of background colors
- `lighten` backgrounds are replaced with the lightest of the background colors

Several other *type* options are also available. You can specify a comma-separated list of blend values, so that each background layer is blended in a different way with all the other backgrounds.

TRY IT

To explore the impact of the `background-blend-mode` style on multiple backgrounds, open the demo_blend.html file in the html04 ▸ demo folder.

Working with Borders

So far, you have only worked with the content, padding, and margin spaces from the CSS Box model. Now, you will examine the border space that separates the element's content and padding from its margins and essentially marks the extent of the element as it is rendered on the page.

Setting Border Width and Color

CSS supports several style properties that are used to format the border around each element. As with the margin and padding styles, you can apply a style to the top, right, bottom, or left border, or to all borders at once. To define the thickness of a specific border, use the property

```
border-side-width: width;
```

where *side* is either `top`, `right`, `bottom`, or `left` and *width* is the width of the border in one of the CSS units of measure. For example, the following style sets the width of the bottom border to 10 pixels.

```
border-bottom-width: 10px;
```

Border widths also can be expressed using the keywords `thin`, `medium`, or `thick`; the exact application of these keywords depends on the browser. You can define the border widths for all sides at once using the `border-width` property

```
border-width: top right bottom left;
```

where *top*, *right*, *bottom*, and *left* are the widths of the matching border. As with the `margin` and `padding` properties, if you enter one value, it's applied to all four

borders; two values set the width of the top/bottom and left/right borders, respectively; and three values are applied to the top, left/right, and bottom borders, in that order. Thus, the following property sets the widths of the top/bottom borders to 10 pixels and the left/right borders to 20 pixels:

```
border-width: 10px 20px;
```

The color of each individual border is set using the property

```
border-side-color: color;
```

where *side* once again specifies the border side and *color* is a color name, color value, or the keyword `transparent` to create an invisible border. The color of the four sides can be specified using the following `border-color` property

```
border-color: top right bottom left;
```

where *top right bottom left* specifies the side to which the color should be applied. Thus, the following style uses gray for the top and left borders and black for the right and bottom borders:

```
border-color: gray black black gray;
```

If no border color is specified, the border will use the text color assigned to the element.

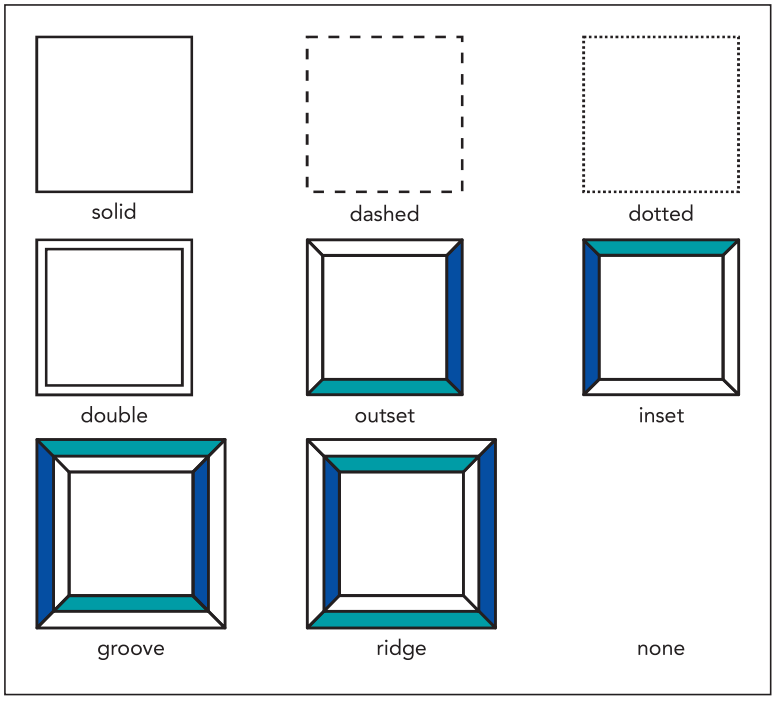
Setting the Border Design

CSS allows you to further define the appearance of borders using the following border styles:

```
border-side-style: style;
```

where *side* once again indicates the border side and *style* specifies one of the nine border styles displayed in Figure 4–12.

Figure 4–12 Examples of border styles



Or to specify styles for all four borders use the property:

```
border-style: top right bottom left;
```

As with the other border rules, you can modify the style of all borders or combinations of the borders. For example, the following style uses a double line for the top/bottom borders and a single solid line for the left/right borders.

```
border-style: double solid;
```

All of the border styles discussed above can be combined into the following property that formats the width, style, and color of all of the borders

```
border: width style color;
```

TRY IT

You can explore the CSS border style using the demo_border.html file in the html04 ▸ demo folder.

where *width* is the thickness of the border, *style* is the style of the border, and *color* is the border color. The following style rule inserts a 2-pixel-wide solid blue border around every side of each h1 heading in the document:

```
h1 {border: 2px solid blue;}
```

To modify the width, style, and color of a single border, use the property

```
border-side: width style color;
```

where *side* is either top, right, bottom, or left.

REFERENCE

Adding a Border

- To add a border around every side of an element, use the CSS property

  ```
  border: width style color;
  ```

  where *width* is the width of the border, *style* is the design style, and *color* is the border color.
- To apply a border to a specific side, use

  ```
  border-side: width style color;
  ```

  where *side* is top, right, bottom, or left for the top, right, bottom, and left borders.
- To set the width, style, or color of a specific side, use the properties

  ```
  border-side-width: width;
  border-side-style: style;
  border-side-color: color;
  ```

Kevin wants the page body to stand out better against the tiled images used as the background for the browser window. He suggests you add solid borders to the left and right edges of the page body and that you add a double border around the aside element containing links to other Komatsu family pages.

To add borders to the page elements:

1. Return to the **tb_visual1.css** file in your editor and go to the Page Body Styles section.
2. Add the following style rule for the page body:

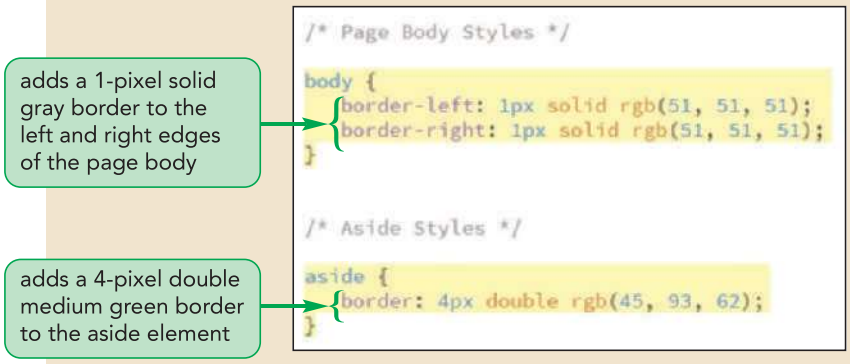
```
body {
   border-left: 1px solid rgb(51, 51, 51);
   border-right: 1px solid rgb(51, 51, 51);
}
```

3. Go to the Aside Styles section and add the following style rule for the aside element:

```
aside {
   border: 4px double rgb(45, 93, 62);
}
```

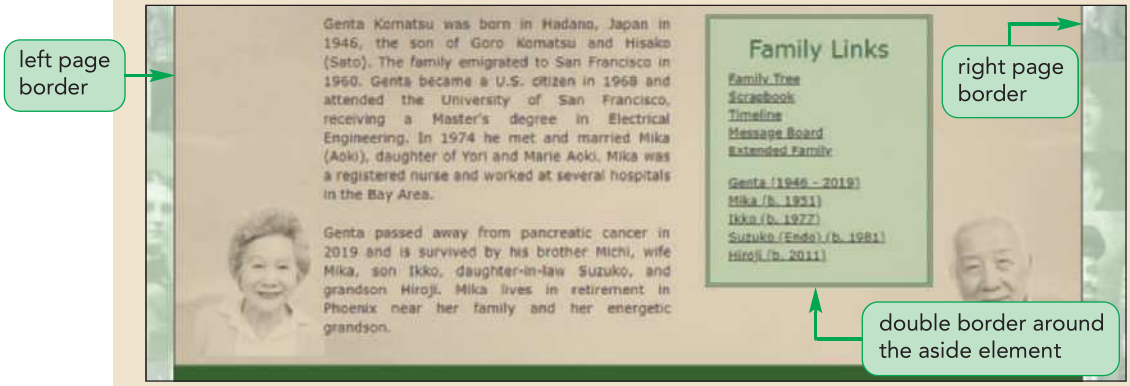
Figure 4–13 highlights the style rules that create borders for the page body and `aside` element.

Figure 4–13 Adding borders to the page body and aside element



4. Save your changes to the file and then reload tb_komatsu.html in your browser. Figure 4–14 shows the appearance of the page with the newly added borders. Note that the background color and other styles associated with the `aside` element are in the tb_styles1.css file.

Figure 4–14 Page design with borders



Source: Wikimedia Commons; © imtmphoto/Shutterstock.com

Kevin is concerned that the design of the page is too boxy and he wants you to soften the design by adding curves to some of the page elements. You can create this effect using rounded corners.

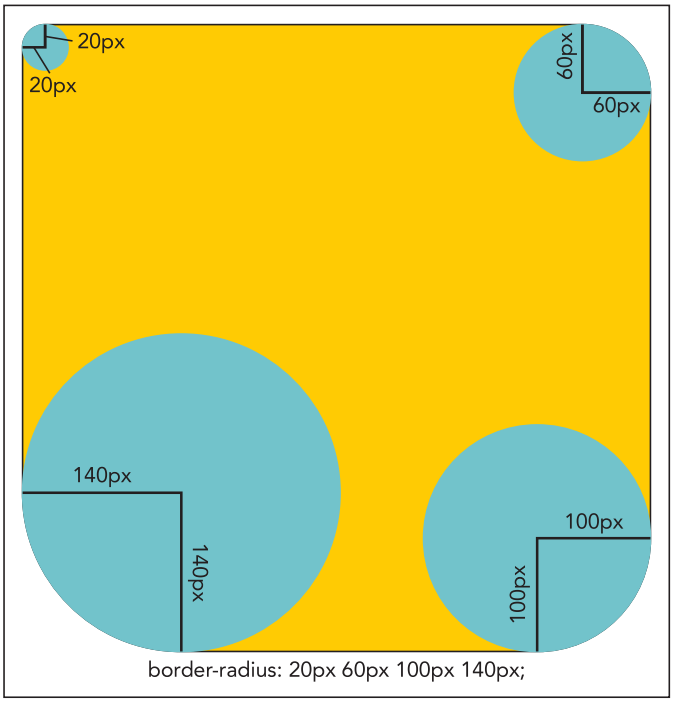
Creating Rounded Corners

To round off any of the four corners of a border, apply the following `border-radius` property:

```
border-radius: top-left top-right bottom-right bottom-left;
```

where *top-left*, *top-right*, *bottom-right*, and *bottom-left* are the radii of the individual corners. The radii are equal to the radii of hypothetical circles placed at the corners of the box with the arcs of the circles defining the rounded corners (see Figure 4–15).

Figure 4–15 Setting rounded corners based on corner radii



TRY IT

You can explore the CSS border-radius style using the demo_radius.html file in the html04 ▸ demo folder.

If you enter only one radius value, it is applied to all four corners; if you enter two values, the first is applied to the top-left and bottom-right corners, and the second is applied to the top-right and bottom-left corners. If you specify three radii, they are applied to the top-left, top-right/bottom-left, and bottom-right corners, in that order. For example, the following style rule creates rounded corners for the `aside` element in which the radii of the top-left and bottom-right corners is 50 pixels and the radii of the top-right and bottom-left corners is 20 pixels.

```
aside {border-radius: 50px 20px;}
```

To set the curvature for only one corner, use the property:

```
border-corner-radius: radius;
```

where *corner* is either `top-left`, `top-right`, `bottom-right`, or `bottom-left`.

REFERENCE

Creating a Rounded Corner

- To create rounded corners for an element border, use

```
border-radius: top-left top-right bottom-right bottom-left;
```

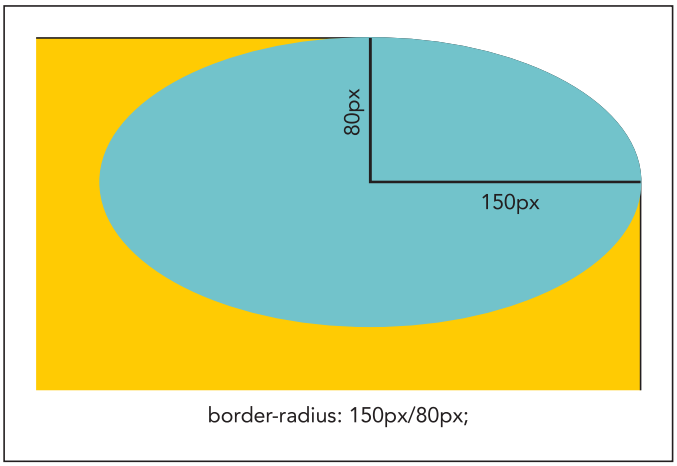
where *radius* is the radius of the rounded corner in one of the CSS units of measurement and *top-left*, *top-right*, *bottom-right*, and *bottom-left* are the radii of the individual corners.

The corners do not need to be circular. Elongated or elliptical corners are created by specifying the ratio of the horizontal radius to the vertical radius using the style:

```
border-radius: horizontal/vertical;
```

where *horizontal* is the horizontal radius of the corner and *vertical* is the vertical radius of the same corner (see Figure 4–16).

Figure 4–16 Creating an elongated corner



Thus, the following style rule creates elongated corners in which the ratio of the horizontal to vertical radius is 50 pixels to 20 pixels.

```
border-radius: 50px/20px;
```

TIP

To create a circular border, use a square element with an equal width and height and the corner radii set to 50%.

Note that using percentages for the radius value can result in elongated corners if the element is not perfectly square. The following style rule sets the horizontal radius to 15% of element width and 15% of the element height. If the element is twice as wide as it is high for example, the corners will not be rounded but elongated.

```
border-radius: 15%;
```

When applied to a single corner, the format to create an elongated corner is slightly different. You remove the slash between the horizontal and vertical values and use the following syntax:

```
border-corner-radius: horizontal vertical;
```

For example, the following style creates an elongated bottom-left corner with a horizontal radius of 50 pixels and a vertical radius of 20 pixels.

```
border-bottom-left-radius: 50px 20px;
```

TRY IT

You can explore how to create elliptical corners using the demo_ellipse.html file in the html04 ▸ demo folder.

Rounded and elongated corners do not clip element content. If the content of the element extends into the corner, it will still be displayed as part of the background. Because this is often unsightly, you should avoid heavily rounded or elongated corners unless you can be sure they will not obscure or distract from the element content.

Add rounded corners with a radius of 30 pixels to the `aside` element.

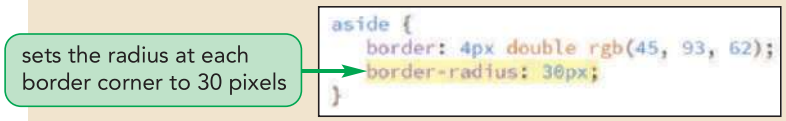
To add rounded corners to an element:

1. Return to the **tb_visual1.css** file in your editor and go to the Aside Styles section.
2. Add the following style to the style rule for the `aside` element:

```
border-radius: 30px;
```

Figure 4–17 highlights the style to create the rounded corners for the aside border.

Figure 4–17 Adding rounded corners to the aside element border



3. Save your changes to the file and reload tb_komatsu.html in your browser. Figure 4–18 shows the rounded corners for the `aside` element border.

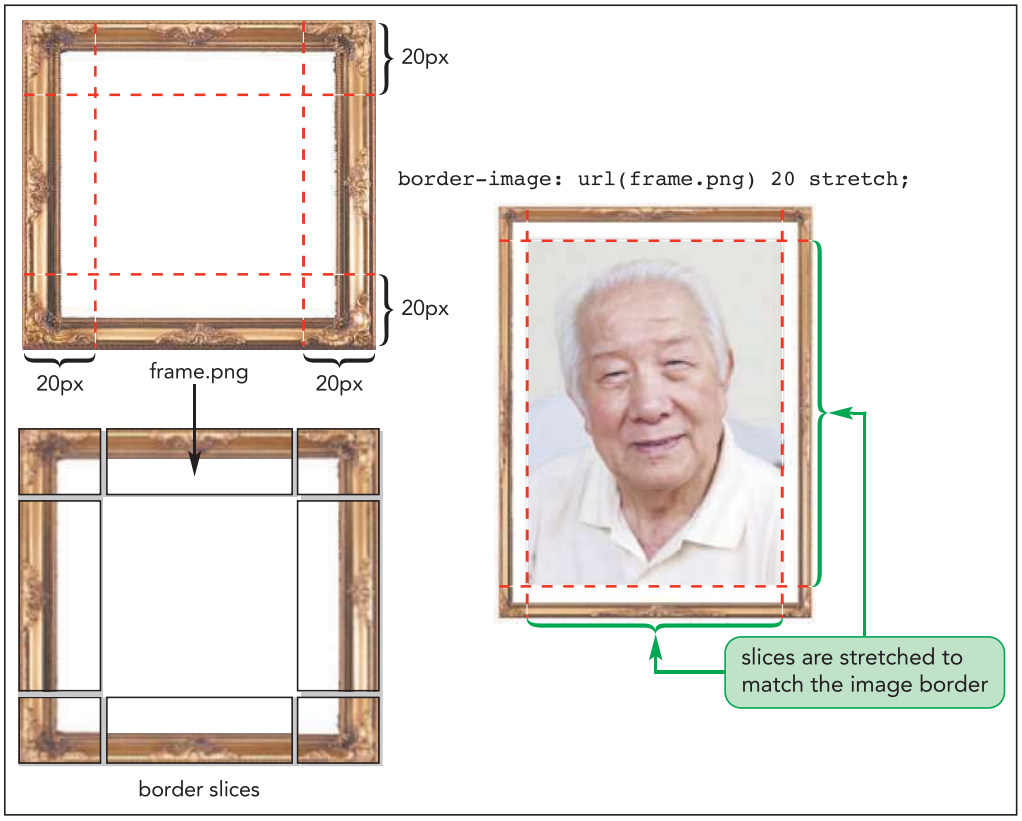
Figure 4–18 Aside element border with rounded corners

Kevin likes the revision to the border for the `aside` element. He also wants you to add a border to the family portrait on the Komatsu Family page. However, rather than using one of the styles shown in Figure 4–12, Kevin wants you to use a graphic border that makes it appear as if the figure box came from a torn piece of paper. You can create this effect using border images.

Applying a Border Image

A border image is a border that it is based on a graphic image. The graphic image is sliced into nine sections representing the four corners, the four sides, and the interior piece. The interior piece is discarded because that is where the content of the object will appear; the four corners become the corners of the border and the four sides are either stretched or tiled to fill in the border's top, right, bottom, and left sides. Figure 4–19 shows an example of an image file, frame.png, sliced into nine sections to create a border image.

Figure 4–19 **Slicing a graphic image to create a border**



To apply a border image, use the following property

```
border-image: url(url) slice repeat fill;
```

where *url* is the source of the graphic image, *slice* is the width or height of the slices used to create the sides and corners, *repeat* indicates whether the side slices should be stretched or tiled to cover the border's four sides, and `fill` is an optional attribute that fills the image background with the graphic image file. The *repeat* option supports the following values:

- `stretch`: The slices are stretched to fill each side.
- `repeat`: The slices are tiled to fill each side.
- `round`: The slices are tiled to fill each side; if they don't fill the sides with an integer number of tiles, the slices are rescaled until they do.
- `space`: The slices are tiled to fill each side; if they don't fill the sides with an integer number of tiles, extra space is distributed around the tiles.

For example, the following style cuts 10-pixel-wide slices from the frame.png image file with the four side slices stretched to cover the length of the four sides of the object's border:

```
border-image: url(frame.png) 10 stretch;
```

The size of the slices is measured either in pixels or as a percentage of the image file width and height. A quirk of this property is that you should *not* specify the pixel unit if you want the slices measured in pixels but you must include the % symbol when slices are measured in percentages.

You can create slices of different widths or heights by entering the size values in a space-separated list. For instance, the following style slices the graphic image 5 pixels on the top, 10 pixels on the right, 15 pixels on the bottom, and 25 pixels on the left:

```
border-image: url(frame.png) 5 10 15 25 stretch;
```

TRY IT

You can explore how to create a border image using the demo_frame.html file in the html04 ▸ demo folder.

The slice sizes follow the same top/right/bottom/left syntax used with all of the CSS border styles. Thus, the following style slices 5% from the top and bottom sides of the graphic image, and 10% from the left and right sides:

```
border-image: url(frame.png) 5% 10% stretch;
```

You can also apply different repeat values to different sides of the border. For example, the following style stretches the border slices on the top and bottom but tiles the left and right slices:

```
border-image: url(frame.png) 10 stretch repeat;
```

REFERENCE

Creating a Graphic Border

- To create a border based on a graphic image, use

```
border-image: url(url) slice repeat fill;
```

where *url* is the source of the border image file, *slice* is the size of the border image cut off to create the borders, *repeat* indicates whether the side borders should be either stretched or tiled to cover the object's four sides, and `fill` is an optional attribute that fills the image background with the graphic image file.

The torn paper image that Kevin wants to use is based on the graphic image file tp_border.png file. Use the `border-image` property to add a border image around the figure box on the Komatsu Family page, tiling the border slices to fill the sides. Note that in order for the border image to appear you must include values for the `border-width` and `border-style` properties.

To create a graphic border:

1. Return to the **tb_visual1.css** file in your editor and scroll to the Figure Box Styles at the top of the file.
2. Add the following style to the style rule for the figure box:

```
border-style: solid;
border-width: 25px;
border-image: url(tb_border.png) 50 repeat;
```

Figure 4–20 displays the styles used to create the graphic border.

Figure 4–20 Adding a border image

3. Save your changes and reload tb_komatsu.html in your browser. Figure 4–21 shows the appearance of the border image.

Figure 4–21 Figure box with border image

Kevin appreciates the effect you created, making it appear as if the family portrait was torn from an album and laid on top of the web page.

PROSKILLS

Problem Solving: Graphic Design and Legacy Browsers

Adding snazzy graphics to your page can be fun, but you must keep in mind that the fundamental test of your design is not how cool it looks but how usable it is. Any design you create needs to be compatible across several browser versions if you want to reach the widest user base. To support older browsers, your style sheet should use progressive enhancement in which the older properties are listed first, followed by browser extensions, and then by the most current CSS properties. As each property supersedes the previous properties, the browser will end up using the most current property that it supports.

For example, the following style rule starts with a basic 5-pixel blue border that will be recognized by every browser. It is followed by browser extensions for Opera, Mozilla, and WebKit to support older browsers that predate adoption of the CSS `border-image` property. Finally, the style list ends with the CSS `border-image` property, recognized by every current browser. In this way, every browser that opens the page will show some type of border.

```
border: 5px solid blue;
-o-border-image: url(paper.png) 30 repeat;
-moz-border-image: url(paper.png) 30 repeat;
-webkit-border-image: url(paper.png) 30 repeat;
border-image: url(paper.png) 30 repeat;
```

Be aware, however, that the syntax for an extension may not match the syntax for the final CSS specification. For example, the following list of styles creates a rounded top-right corner that is compatible across a wide range of browser versions:

```
-moz-border-radius-top-right: 15px;
-webkit-border-top-right-radius: 15px;
border-top-right-radius: 15px;
```

Note that the syntax for the Mozilla extension does not match the syntax for the WebKit extension or for the final CSS specification. As always, you need to do your homework to learn exactly how different browser versions handle these CSS design styles.

In the next session, you'll continue to work with the CSS graphic styles to add three-dimensional effects through the use of drop shadows and color gradients. If you want to take a break, you can close your open files and documents now.

REVIEW

Session 4.1 Quick Check

1. The element to create a figure caption is:
   a. `caption`
   b. `figurecap`
   c. `figcaption`
   d. `alt`
2. Lines and curves based on mathematical functions comprise a:
   a. bitmap image
   b. png image
   c. figure image
   d. vector image
3. To tile a background image in the horizontal direction only, use:
   a. `repeat-x`
   b. `repeat-y`
   c. `repeat-horizontal`
   d. `repeat-h`
4. To display the background image only within a content box, apply the style:
   a. `background-origin: content-box;`
   b. `background: content-box;`
   c. `background-display: content-box;`
   d. `background-clip: content-box;`
5. To create a 5-pixel wide brown border with a dotted line, apply the style:
   a. `border-style: 5px brown dotted;`
   b. `border-type: brown 5px dotted;`
   c. `border-outline: dotted brown 5px;`
   d. `border: 5px brown dotted;`
6. To use rounded corners with a radius of 15 pixels in a border, apply the style:
   a. `border-width: 15px;`
   b. `border-radius: 15px;`
   c. `border-arc: 15px;`
   d. `corner-radius: 15px;`
7. To create an elongated corner with a horizontal radius of 10 pixels and a vertical radius of 5 pixels, use:
   a. `border-radius: 10px 5px;`
   b. `border-radius: 5px 10px;`
   c. `border-radius: 10px/5px;`
   d. `border-radius: 5px/10px;`
8. To create a border image using the border.png file with a slice size of 30 pixels and the slices stretched along the borders, use:
   a. `border-image: url(border.png) 30 stretch;`
   b. `border: url(border.png) 30 stretch;`
   c. `border-img: url(border.png) 30 stretch;`
   d. `border-slice: url(border.png) 30 stretch;`

Session 4.2 Visual Overview:

The color value sets the shadow color.

The distance values set the shadow offsets.

This value sets the shadow blur.

The **box-shadow** property adds a drop shadow to a block element.

The **inset** keyword places the shadow inside the element.

The **text-shadow** property adds a drop shadow to a text string.

The **radial-gradient** function creates a color gradient proceeding outward from a central point.

The **linear-gradient** function creates a color gradient proceeding along a straight line.

The **opacity** property makes an object semi-transparent.

This value sets the shadow size.

These values set the shadow offsets.

These color values set the shadow color.

This value sets the shadow blur.

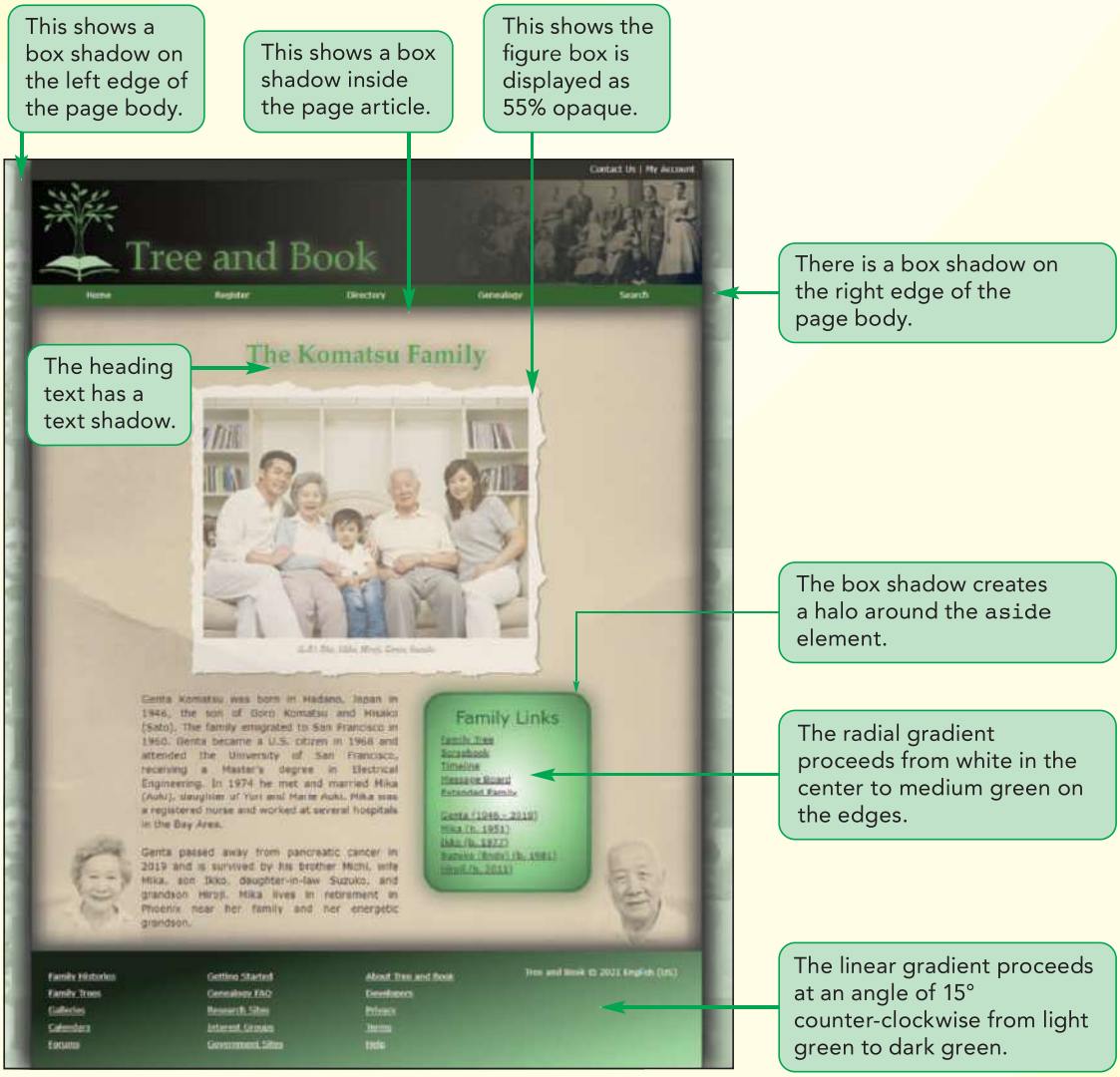
The **color-stop** defines the extent of a color within a gradient.

This value sets the opacity of the figure to 55%.

```
/* Shadow Styles */
body {
   box-shadow: rgb(51, 51, 51) 15px 0px 25px,
               rgb(51, 51, 51) -15px 0px 25px;
}

article {
   box-shadow: inset rgb(71, 71, 71) -10px -10px 25px,
               inset rgb(71, 71, 71) 10px 10px 25px;
}

aside {
   box-shadow: rgba(51, 91, 51, 0.4) 0px 0px 20px 10px;
}

article header h1 {
   text-shadow: rgb(181, 211, 181) 2px 2px 1px,
                rgba(21, 21, 21, 0.66) 5px 5px 25px;
}

/* Gradient Styles */
aside {
   background: radial-gradient(white, rgb(151, 222, 151), rgb(81, 125, 81));
}

footer {
   background: linear-gradient(345deg, rgb(172, 232, 172), rgb(21, 35, 21) 80%);
}

/* Opacity Styles */
body > article > header > figure {
   opacity: 0.55;
}
```

Logo Design Studio Pro; Source: wiki Media;

Shadows and Gradients



Source: Design Studio Pro; Source: Wikimedia Commons; imtmphoto/Shutterstock.com

Creating Drop Shadows

In this session, you will examine some design styles that create 3D effects, making the page content appear to jump out of the browser window. The first styles you'll explore are used to create drop shadows around text strings and element boxes.

Creating a Text Shadow

To give the text on your page visual impact, you can use CSS to add a shadow using the following `text-shadow` property

```
text-shadow: color offsetX offsetY blur;
```

where *color* is the shadow color, *offsetX* and *offsetY* are the distances of the shadow from the text in the horizontal and vertical directions, and *blur* defines the amount by which the shadow spreads out, creating a blurred effect. The shadow offset values are expressed so that positive values push the shadow to the right and down while negative values move the shadow to the left and up. The default *blur* value is 0, creating a shadow with distinct hard edges; as the blur value increases, the edge of the shadow becomes less distinct and blends more in the text background.

The following style creates a red text shadow that is 10 pixels to the right and 5 pixels down from the text with blur of 8 pixels:

```
text-shadow: red 10px 5px 8px;
```

Multiple shadows can be added to text by including each shadow definition in the following comma-separated list.

```
text-shadow: shadow1, shadow2, shadow3, …;
```

where *shadow1*, *shadow2*, *shadow3*, and so on are shadows applied to the text with the first shadow listed displayed on top of subsequent shadows when they overlap. The following style rule creates two shadows with the first red shadow placed 10 pixels to the left and 5 pixels up from the text and the second gray shadow is placed 3 pixels to the right and 4 pixels down from the text. Both shadows have a blur of 6 pixels:

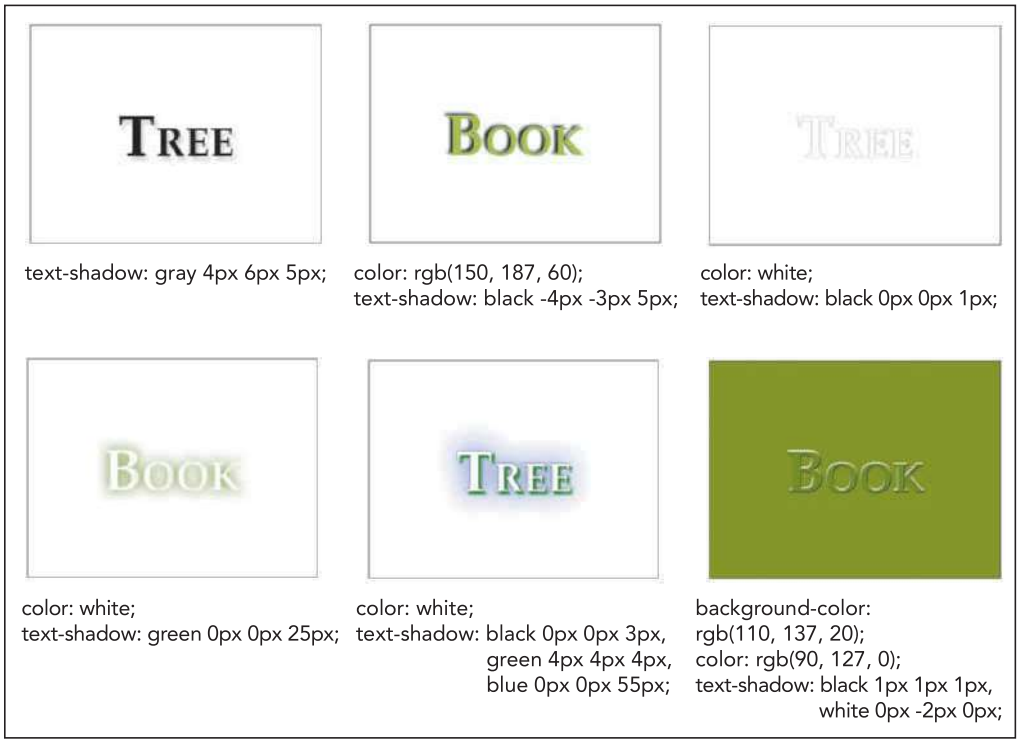
```
text-shadow: red -10px -5px 6px,
             gray 3px 4px 6px;
```

TRY IT

You can explore the text-shadow style and creating multiple text shadows by using the demo_text.html file from the html04 ▸ demo folder.

Figure 4–22 shows examples of how the `text-shadow` style can be used to achieve a variety of text designs involving single and multiple shadows.

Figure 4–22 Examples of text shadows



REFERENCE

Creating a Text Shadow

- To add a shadow to a text string, use the property

```
text-shadow: color offsetX offsetY blur;
```

where *color* is the shadow color, *offsetX* and *offsetY* are the distances of the shadow from the text in the horizontal and vertical directions, and *blur* defines the amount by which the shadow is stretched.

Kevin wants you to add two text shadows to the h1 heading *The Komatsu Family*. The first text shadow will be a light-green highlight with hard edges and the second shadow will be semi-transparent gray and blurred.

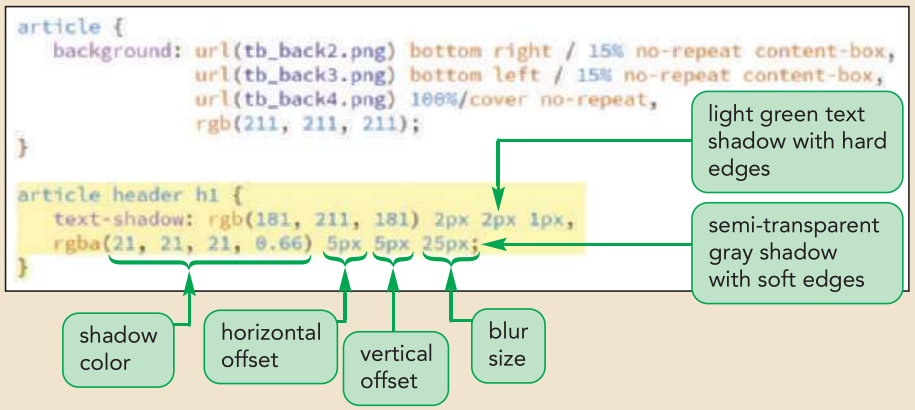
To add a text shadow:

1. If you took a break after the previous session, reopen or return to the **tb_visual1.css** file in your editor and scroll to the Article Styles section.
2. Add the following style for the h1 heading in the article header:

```
article header h1 {
   text-shadow: rgb(181, 211, 181) 2px 2px 1px,
                rgba(21, 21, 21, 0.66) 5px 5px 25px;
}
```

Figure 4–23 highlights the style to add text shadows to the h1 heading.

Figure 4–23 Adding text shadows



3. Save your changes and reload tb_komatsu.html in your browser. Figure 4–24 shows the shadow effect added to the h1 heading.

Figure 4–24 Article heading with text shadows

© imtmphoto/Shutterstock.com

Kevin likes the shadow effect and the use of the light green shadow, which appears to give a highlight to the heading text. Next, he wants you to add shadows to other page objects.

Creating a Box Shadow

Shadows can be added to any block element in the web page by using the `box-shadow` property

```
box-shadow: color offsetX offsetY blur;
```

where *color*, *offsetX*, *offsetY*, and *blur* have the same meanings for box shadows as they do for text shadows. As with text shadows, you can add multiple shadows by including them in the following comma-separated list

```
box-shadow: shadow1, shadow2 …;
```

where once again the first shadow listed is displayed on top of subsequent shadows.

TIP

If no shadow color is provided, the browser uses black as the default color.

In the last session, you used left and right borders to set off the page body from the browser window background. Kevin would like you to increase this visual distinction by adding drop shadows to the left and right sides of the page body.

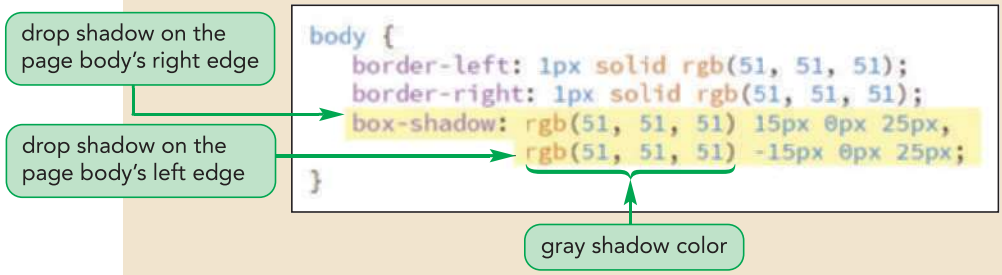
To add a box shadow:

1. Return to the **tb_visual1.css** file in your editor and go to the Page Body Styles section.
2. Within the style rule for the `body` element, insert the following styles:

```
box-shadow: rgb(51, 51, 51) 15px 0px 25px,
            rgb(51, 51, 51) -15px 0px 25px;
```

Figure 4–25 highlights the style to add box shadows to the page body.

Figure 4–25 Adding box shadows



3. Save your changes and reload tb_komatsu.html in your browser. Figure 4–26 shows the drop shadows added to the page body.

Figure 4–26 Page body with drop shadows

Source: Design Studio Pro; Source: Wikimedia Commons; © imtmphoto/Shutterstock.com

Box shadows can be placed inside the element as well as outside. By adding an interior shadow you can create the illusion of a beveled edge in which the object appears to rise out of its background. To create an interior shadow, add the `inset` keyword to the `box-shadow` property

```
box-shadow: inset color offsetX offsetY blur;
```

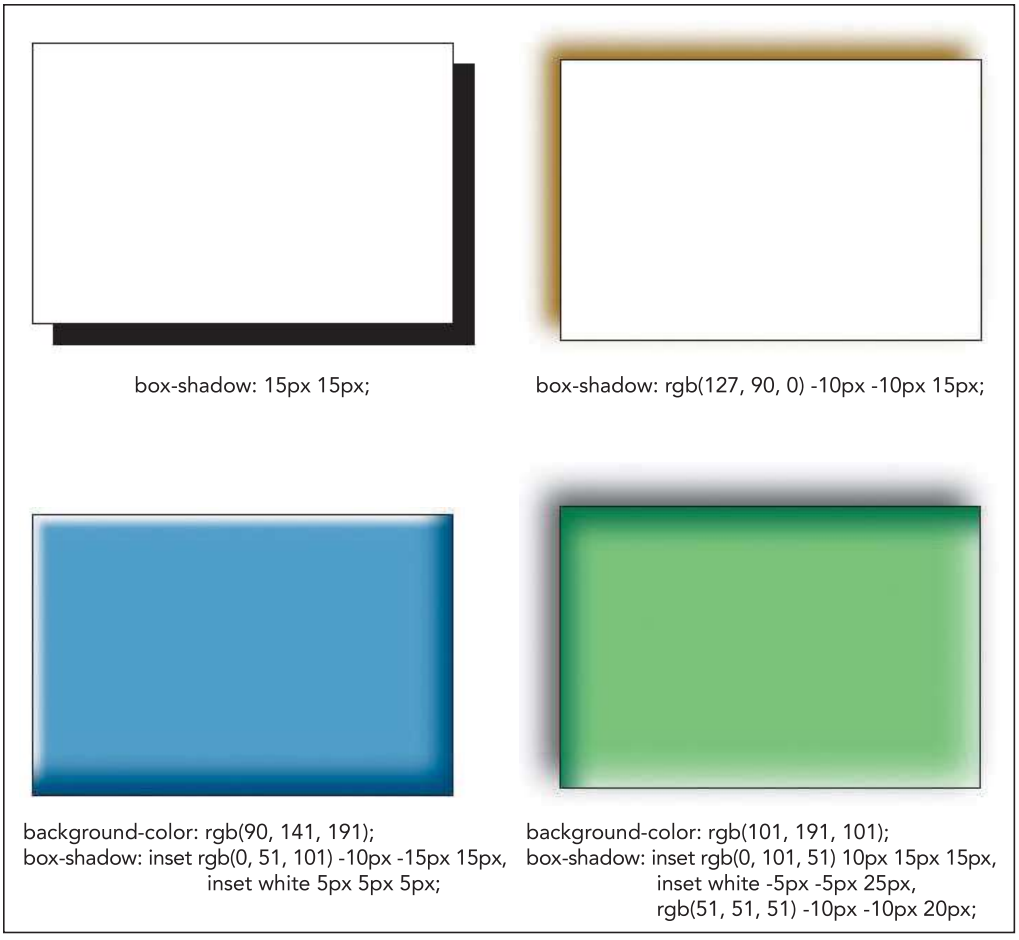
where the meanings of the *offsetX* and *offsetY* values are switched when applied to interior shadowing so that positive *offsetX* and *offsetY* values move the shadow to the left and up within the box, while negative *offsetX* and *offsetY* values move the shadow to the right and down.

TRY IT

Explore multiple box shadows with the demo_box.html file from the html04 ▸ demo folder.

An object can contain a mixture of exterior and interior shadows. Figure 4–27 shows examples of box shadows, including one example that mixes both interior and exterior shadows.

Figure 4–27 Examples of box shadows



Kevin suggests that you add inset shadows to the `article` element, placing medium gray shadows within the article to make it appear raised up from the surrounding page content.

Positive and negative offset values for interior shadows have the opposite meaning from positive and negative offset values for exterior shadows.

To add inset shadows:

1. Return to the **tb_visual1.css** file in your editor and go to the Article Styles section.
2. Within the style rule for the `article` element, insert the following `box-shadow` style:

```
box-shadow: inset rgb(71, 71, 71) -10px -10px 25px,
            inset rgb(71, 71, 71) 10px 10px 25px;
```

Figure 4–28 highlights the newly added code for the inset box shadow.

Figure 4–28 Adding an inset shadow

places a medium-gray shadow in the lower-right interior corner

inset keyword places shadow inside the object

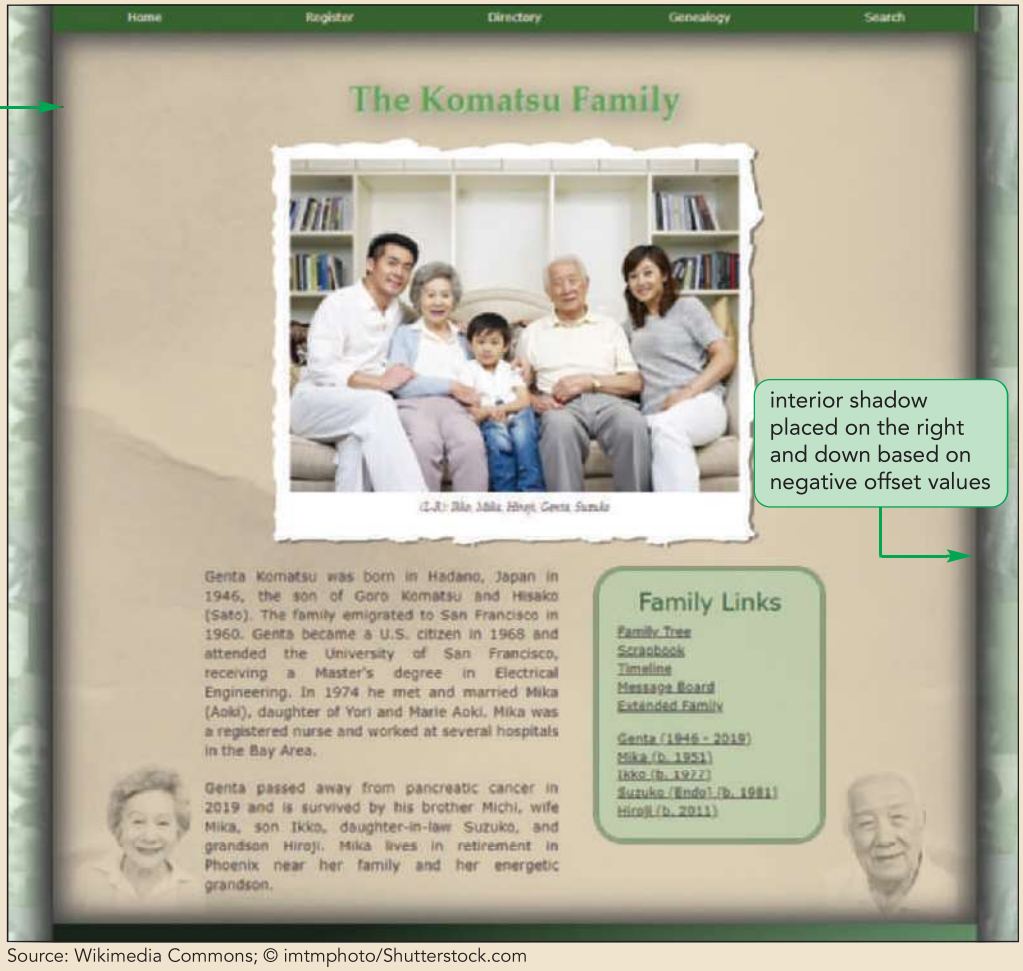
```
article {
   background: url(tb_back2.png) bottom right / 15% no-repeat content-box,
               url(tb_back3.png) bottom left / 15% no-repeat content-box,
               url(tb_back4.png) 100%/cover no-repeat,
               rgb(211, 211, 211);
   box-shadow: inset rgb(71, 71, 71) -10px -10px 25px,
               inset rgb(71, 71, 71) 10px 10px 25px;
}
```

places a medium-gray shadow in the upper-left interior corner

3. Save your changes and reload tb_komatsu.html in your browser. The inset shadow for the page body element is shown in Figure 4–29.

Figure 4–29 Page article with interior shadowing

interior shadow placed on the left and up based on positive offset values



interior shadow placed on the right and down based on negative offset values

Source: Wikimedia Commons; © imtmphoto/Shutterstock.com

By default, a box shadow has the same size and dimensions as its page object offset in the horizontal and vertical direction. To change the shadow size, add the *spread* parameter to the `box-shadow` property, specifying the size of the shadow relative to the size of the page object. A positive value increases the size of the shadow, while a negative value decreases it. For example, the following style creates a gray shadow that is offset from the page object by 5 pixels in both the vertical and horizontal direction

with no blurring but with a shadow that is 15 pixels larger in the horizontal and vertical directions than the object:

```
box-shadow: gray 5px 5px 0px 15px;
```

On the other hand, the following style creates a shadow that is 15 pixels smaller than the page object:

```
box-shadow: gray 5px 5px 0px -15px;
```

REFERENCE

Creating a Box Shadow

- To add a shadow to a block element, use

  ```
  box-shadow: color offsetX offsetY blur spread;
  ```

  where *color* is the shadow color, *offsetX* and *offsetY* are the distances of the shadow from the element in the horizontal and vertical directions, *blur* defines the amount by which the shadow is stretched and *spread* sets the size of the shadow relative to the size of the block element. If no *spread* is specified, the shadow has the same size as the block element.
- To create an interior shadow, include the `inset` keyword

  ```
  box-shadow: inset color offsetX offsetY blur spread;
  ```
- To create multiple shadows place them in a comma-separated list:

  ```
  box-shadow: shadow1, shadow2, …;
  ```

  where *shadow1*, *shadow2*, and so on are definitions for individual shadows with the first shadows listed displayed on top of subsequent shadows.

One application of the *spread* parameter is to create a visual effect in which the object appears to be surrounded by a halo. This is achieved by setting the shadow offsets to 0 pixels while making the shadow larger than the page object itself. Kevin suggests that you use this technique to add a green halo to the `aside` element.

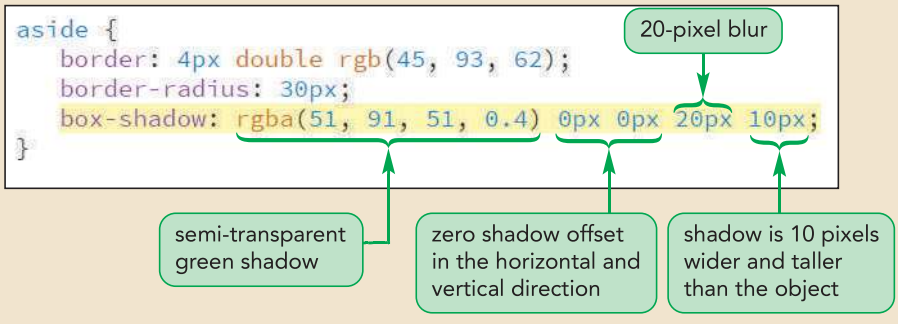
To increase the shadow size:

1. Return to the **tb_visual1.css** file in your editor and go to the Aside Styles section.
2. Within the style rule for the `aside` element, insert the following style:

   ```
   box-shadow: rgba(51, 91, 51, 0.4) 0px 0px 20px 10px;
   ```

   Figure 4–30 highlights the style to add a halo to the `aside` element.

Figure 4–30 Creating a spreading shadow



3. Save your changes and reload tb_komatsu.html in your browser. Figure 4–31 shows the revised appearance of the `aside` element with the glowing green shadow.

Figure 4–31 Aside element with glowing effect

INSIGHT

Creating a Reflection

WebKit, the rendering engine for Safari and Google Chrome, includes support for adding reflections to page objects through the following property

```
-webkit-box-reflect: direction offset mask-box-image;
```

where *direction* is the placement of the reflection using the keywords `above`, `below`, `left`, or `right`; *offset* is the distance of the reflection from the edge of the element box, and *mask-box-image* is an image that can be used to overlay the reflection. For example, the following style rule creates a reflection that is 10 pixels below the inline image:

```
img {
   -webkit-box-reflect: below 10px;
}
```

There is no equivalent `reflect` property in the official W3C CSS specifications. Before using the `reflect` property, you should view the current browser support for the `-webkit-box-reflect` property at *caniuse.com*.

Applying a Color Gradient

So far you have worked with backgrounds consisting of a single color, though that color can be augmented through the use of drop shadows. Another way to modify the background color is through a **color gradient** in which one color gradually blends into another color or fades away if transparent colors are used. CSS supports linear gradients and radial gradients.

Creating a Linear Gradient

A linear gradient is a color gradient in which the background color transitions from a starting color to an ending color along a straight line. Linear gradients are created using the `linear-gradient` function

```
linear-gradient(color1, color2, …)
```

where *color1*, *color2*, and so on are the colors that blend into one another starting from *color1*, through *color2*, and onto the last color listed. The default direction for a linear color gradient is vertical, starting from the top of the object and moving to the bottom.

TIP

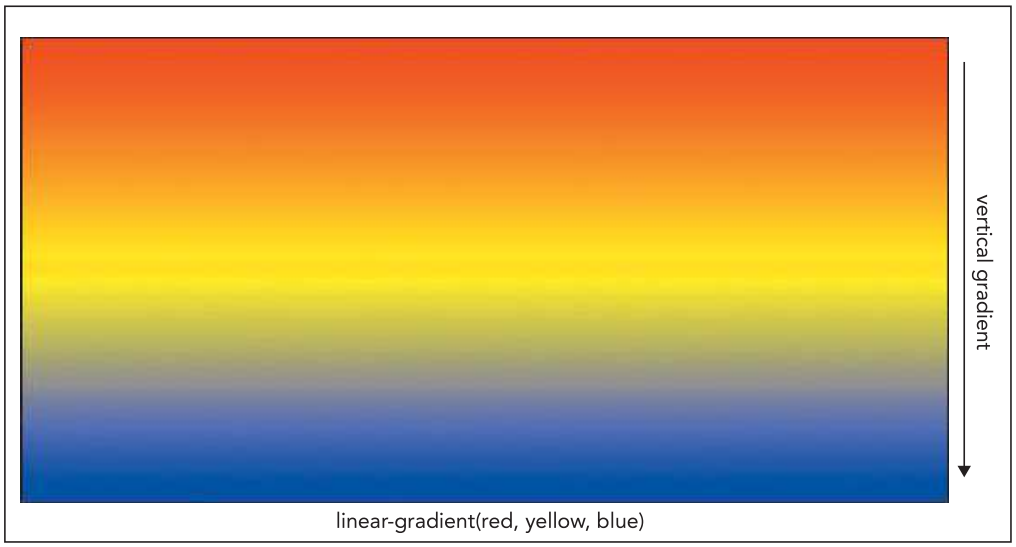
When using multiple backgrounds, gradients can be combined with solid colors and background images to create interesting visual effects; one gradient can also be overlaid on top of another.

Gradients are treated like background images and thus can be used with any CSS property that accepts an image such as the `background`, `background-image`, and `list-style-image` properties. For example, to create a linear gradient as a background for the page body, you could apply the following style rule:

```
body {
   background: linear-gradient(red, yellow, blue);
}
```

Figure 4–32 shows the appearance of this vertical gradient as the background color transitions gradually from red down to yellow and then from yellow down to blue.

Figure 4–32 Linear gradient with three colors



To change from the default vertical direction, you add a *direction* value to the `linear-gradient` function

```
linear-gradient(direction, color1, color2, …)
```

where *direction* is the direction of the gradient using keywords or angles. Direction keywords are written in the form `to` *position* where *position* is either a side of the

object or a corner. For example the following linear gradient moves in a straight line to the left edge of the object blending from red to yellow to blue:

```
background: linear-gradient(to left, red, yellow, blue);
```

To move toward the corner, include both corner edges. The following style moves the gradient in the direction of the object's bottom right corner:

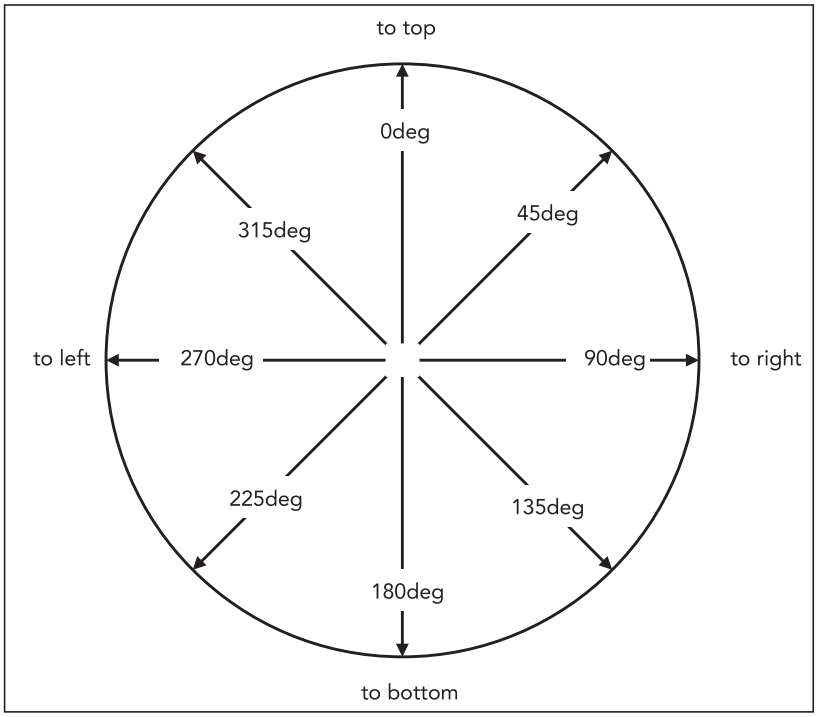
```
background: linear-gradient(to bottom right, red, yellow, blue);
```

TIP

For square objects, a direction of `45deg` is equivalent to a direction of `to right top`.

To move in a direction other than a side or corner, you can express the direction using an angle value. Angles are measured in degrees with 0deg equal to `to top`, 90deg equal to `to right`, 180deg equal to `to bottom`, and 270deg equal to `to left` (see Figure 4–33.)

Figure 4–33 Linear gradient directions



For example, the following gradient points at a 60 degree angle:

```
background: linear-gradient(60deg, red, yellow, blue);
```

Figure 4–34 shows other examples of linear gradients moving in different directions using both syntaxes.

INSIGHT

Transparency and Gradients

Interesting gradient effects can be achieved using transparent colors so that the background color gradually fades away as it moves in the direction of the gradient. For example, the following style creates a linear gradient that gradually fades away from its initial solid red color:

```
linear-gradient(rgba(255, 0, 0, 1), rgba(255, 0, 0, 0))
```

Note that since the final color is completely transparent it will adopt the background color of the parent element.

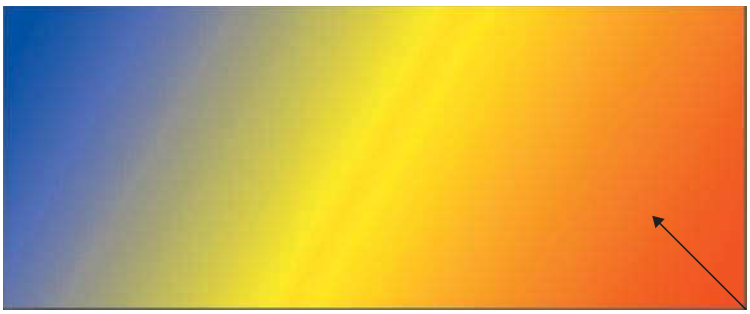
You can also use gradients to create background images that appear to fade by using multiple backgrounds in which the gradient appears on top of an image. For example, the following background style creates a fading background using the back.png image file:

```
background: linear-gradient(rgb(255, 255, 255, 0), rgb(255,
255, 255, 1)),url(back.png));
```

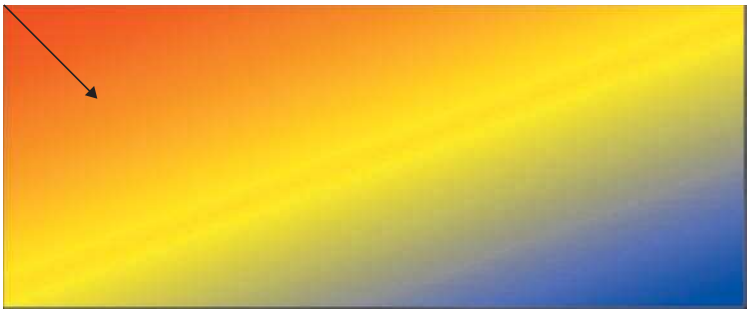
When rendered by the browser, the background image will start as solid but gradually fade to white as the linear gradient proceeds through the element background.

Figure 4–34 Directions of linear gradients

linear-gradient(to left bottom, red, yellow, blue)



linear-gradient(300deg, red, yellow, blue)



linear-gradient(-200deg, red, yellow, blue)

Note that the degree values can be negative in which case the direction is pointed counter-clockwise around the circle shown in Figure 4–33. A negative angle of –45deg, for example, would be equivalent to a positive angle of 315deg, an angle of –200deg would be equal to 160deg, and so forth.

Gradients and Color Stops

The colors specified in a gradient are evenly distributed so that the following gradient starts with a solid red, solid green appears halfway through the gradient, and finishes with solid blue:

```
background: linear-gradient(red, green, blue);
```

To change how the colors are distributed, you define color stops, which represent the point at which the specified color stops and the transition to the next color begins. The `linear-gradient` function using color stops has the general form

```
linear-gradient(direction, color-stop1, color-stop2, …)
```

where *color-stop1*, *color-stop2*, and so on are the colors and their stopping positions within the gradient. Stopping positions can be entered using any of the CSS units of measurement. For example, the following gradient starts with solid red up until 50 pixels from the starting point, red blends to solid green stopping at 60 pixels from the starting point and then blends into solid blue 80 pixels from the start. After 80 pixels, the gradient will remain solid blue to the end of the background.

```
linear-gradient(red 50px, green 60px, blue 80px)
```

TRY IT

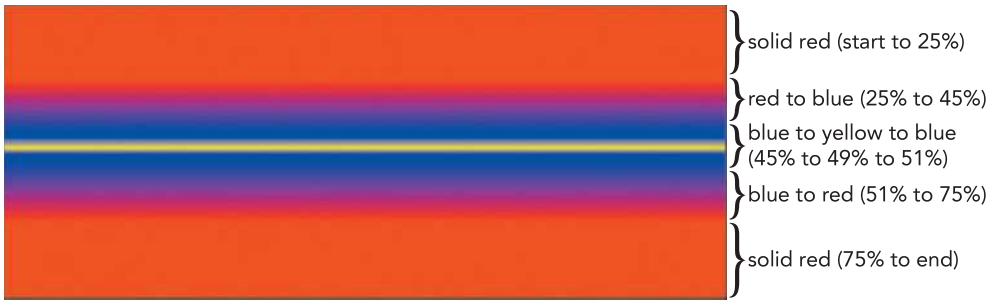
You can create your own linear gradients using the demo_linear.html file from the html04 ▸ demo folder.

Similarly, the following style rule sets the color stops using percentages with solid red for the first 25% of the background, transitioning to solid green from 25% to 75% of the background, and then transitioning to solid blue from 75% to 95% of the background size. From that point to the end, the background remains solid blue.

```
linear-gradient(red 25%, green 75%, blue 95%)
```

Figure 4–35 shows an example of a linear gradient in which color stops are used to create a narrow strip of yellow within a background of red blended into blue.

Figure 4–35 Linear gradient color stops



Kevin suggests you use a linear gradient that transitions from light green to dark green as the background for the page footer.

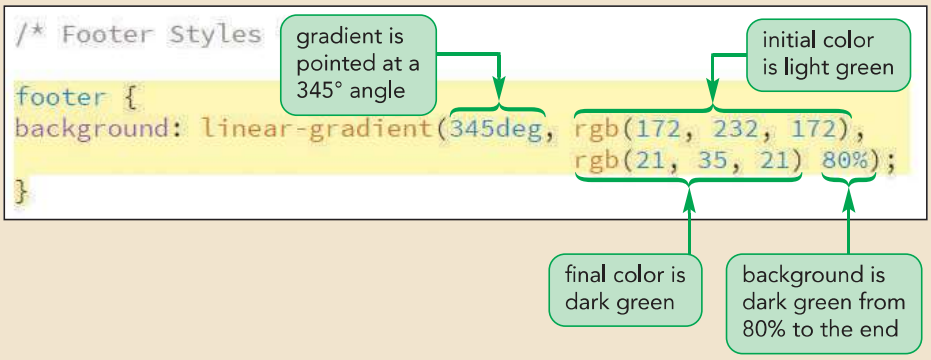
To apply a linear gradient:

1. Return to the **tb_visual1.css** file in your editor and go to the Footer Styles section.
2. Insert the following style rule for the `footer` element:

```
footer {
   background: linear-gradient(345deg, rgb(172, 232, 172),
                                       rgb(21, 35, 21) 80%);
}
```

Figure 4–36 highlights the style to create the linear gradient.

Figure 4–36 Applying a linear gradient



3. Save your changes and reload tb_komatsu.html in your browser. Figure 4–37 shows the revised appearance of the page footer with a linear gradient.

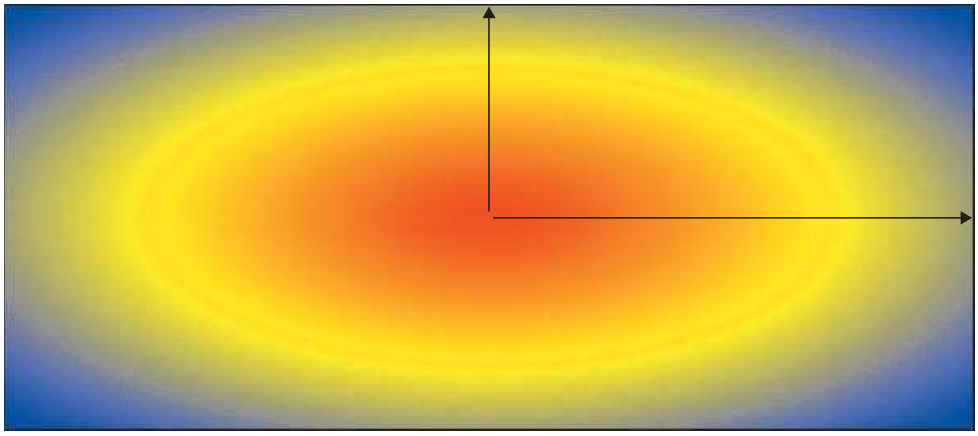
Figure 4–37 Page footer with linear gradient background

The other color gradient supported in CSS is a radial gradient. You will explore how to create radial gradients now.

Creating a Radial Gradient

A **radial gradient** is a color gradient that starts from a central point and proceeds outward in a series of concentric circles or ellipses. Figure 4–38 shows an example of a radial gradient consisting of a series of concentric ellipses radiating from a central red color to an ending blue color.

Figure 4–38 A radial gradient of three colors



radial-gradient(red, yellow, blue)

Radial gradients are created using the following `radial-gradient` function.

```
radial-gradient(shape size at position, color-stop1,
color-stop2, …)
```

The *shape* value defines the shape of the gradient and is either `ellipse` (the default) or `circle`. The *size* value defines the extent of the gradient as it radiates outward and can be expressed with a CSS unit of measure, a percentage of the background's width and height, or with one of the following keywords:

- `farthest-corner` (the default) Gradient extends to the background corner farthest from the gradient's center.
- `farthest-side` Gradient extends to background side farthest from the gradient's center.
- `closest-corner` Gradient extends to the nearest background corner.
- `closest-side` Gradient extends to the background side closest to the gradient's center.

The `position` defines where the gradient radiates from and can be expressed in coordinates using pixels, percentages of the element's width and height, or with the keywords: `left`, `center`, `right`, `top`, and `bottom`. The default is to place the gradient within the center of the background.

Finally the *color-stop1*, *color-stop2* … values are the colors and their stopping positions within the gradient and have the same interpretation used for linear gradients except they mark stopping points as the gradient radiates outward. Note that the color stops are optional, just as they are in linear gradients. For example the following function defines a circular gradient radiating from the horizontal and vertical center of the background through the colors red, yellow, and blue:

```
radial-gradient(circle closest-corner at center center,
                red, yellow, blue)
```

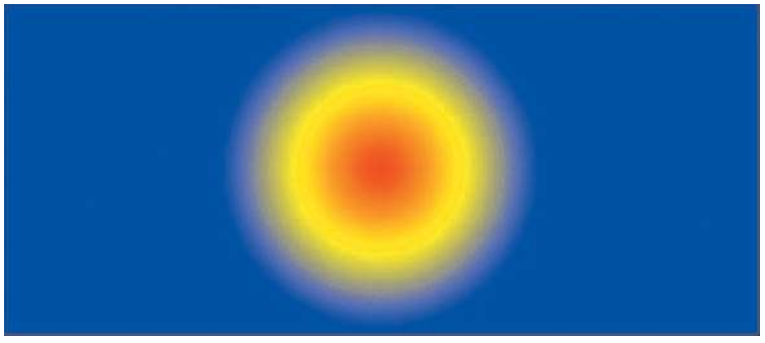
The gradient ends when it reaches the closest background corner. Anything outside of the gradient will be a solid blue.

TRY IT

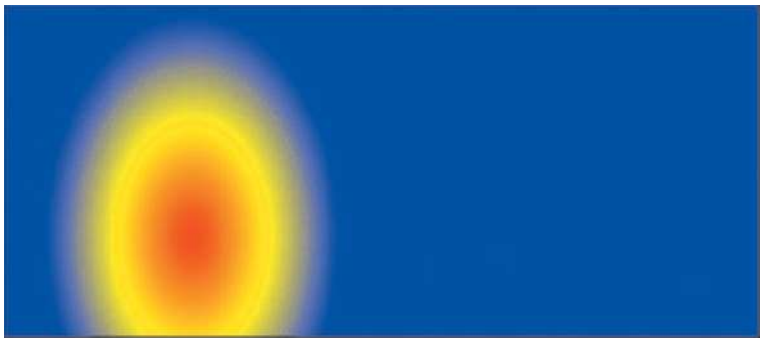
You can explore how to create your own radial gradients using the demo_radial.html file from the html04 ▸ demo folder.

Figure 4–39 shows other examples of the different effects that can be accomplished using the `radial-gradient` function. Note that when parameters of the radial-gradient function are omitted they take their default values.

Figure 4–39 Examples of radial gradients



radial-gradient(circle closest-side, red, yellow, blue)



radial-gradient(120px 180px at 25% 75%, red, yellow, blue)

radial-gradient(at right, red 46%, yellow 50%, blue 54%)

Kevin would like you to apply a radial gradient to the background of the `aside` element. The gradient will start from a white center blending into to a medium green and then into a darker shade of green.

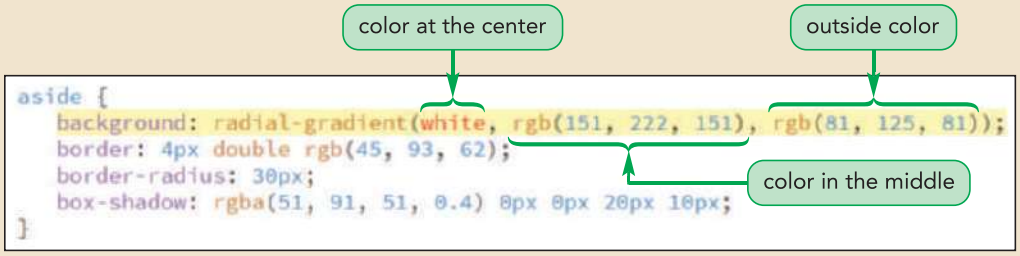
To apply a radial gradient:

1. Return to the **tb_visual1.css** file in your editor and go to the Aside Styles section.
2. Add the following style to the style rule for the `aside` element:

```
background:
radial-gradient(white, rgb(151, 222, 151),
                rgb(81, 125, 81));
```

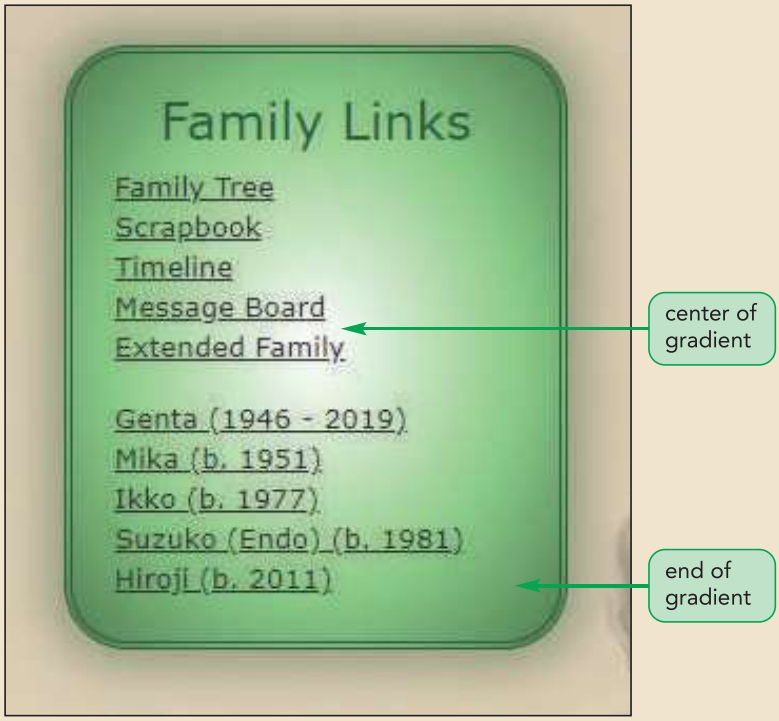
Note that this style supersedes the previous background style created in the tb_styles1.css style sheet. Figure 4–40 highlights the code to create the radial gradient.

Figure 4–40 Applying a radial gradient



3. Save your changes and reload tb_komatsu.html in your browser. Figure 4–41 shows the radial gradient within the `aside` element.

Figure 4–41 Aside element with radial gradient background



Kevin likes the effect of the radial gradient on the `aside` element and feels that it works well with the glowing effect you added earlier.

INSIGHT

Gradients and Browser Extensions

The gradient functions were heavily revised as they went from being browser-specific properties to the final syntax approved by the W3C. If you work with older browsers, you may need to accommodate their versions of these gradient functions. For example, the following linear gradient that blends red to blue going in the direction to the right edge of the background

```
linear-gradient(to right, red, blue)
```

would be expressed using the old WebKit gradient function as:

```
-webkit-gradient(linear, left, right, from(red), to(blue))
```

Other older versions of browsers such as Mozilla, Internet Explorer, and Opera have their own gradient functions with different syntax. You can study these functions using the online support at the browser websites or doing a search on the Web for CSS gradient functions.

Note that not all browser extensions support the same types of gradients, which means that it is difficult and sometimes impossible to duplicate a particular gradient background for every browser. Thus, you should not make gradients an essential feature of your design if you want to be compatible with older browsers.

Repeating a Gradient

As you add more color stops, the gradient function can become unwieldy and overly complicated. One alternative is to repeat the gradient design. You can repeat linear and radial gradients using the functions

```
repeating-linear-gradient(params)
repeating-radial-gradient(params)
```

where *params* are the parameters of the `linear-gradient` or the `radial-gradient` functions already discussed. The only requirement for a repeating gradient is that a stopping position is required for the last color in the list that is less than the size of the object background. Once the last color in the color list is reached, the gradient starts over again. For example, the following function repeats a vertical gradient starting with white transitioning to black, transitioning back to white at 10% of the height of the object, and then repeating that pattern each time it reaches the next 10% of the height of the object:

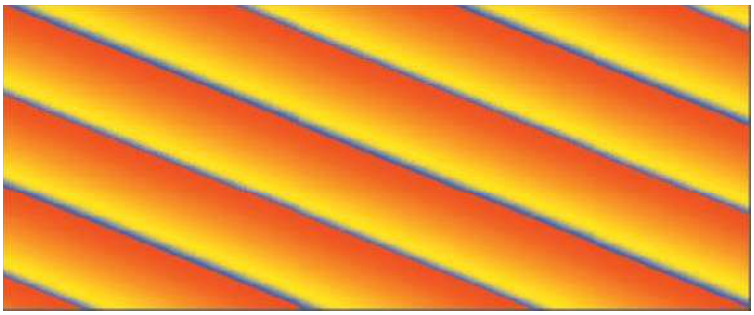
```
repeating-linear-gradient(white, black 10%)
```

Figure 4–42 shows some other examples of repeating linear and radial gradients.

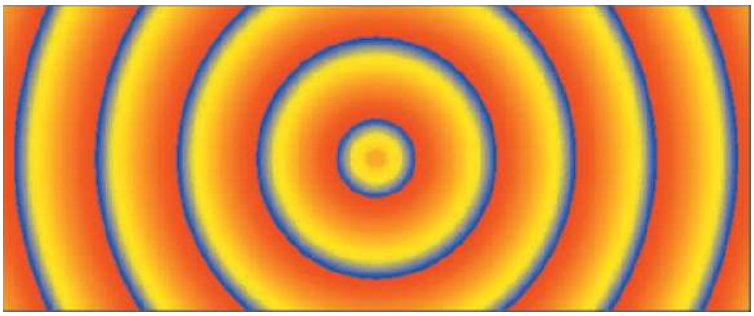
TRY IT

You can create your own repeating gradients using the demo_repeat_linear.html and demo_repeat_radial.html files from the html04 ▸ demo folder.

Figure 4–42 Repeating a gradient



repeating-linear-gradient(to left bottom, red 5%, yellow 18%, blue 20%)



repeating-radial-gradient(circle, red 10%, yellow 25%, blue 30%)

REFERENCE

Creating a Gradient

- To create a linear gradient, use the function

  `linear-gradient(`*`direction`*`, `*`color-stop1`*`, `*`color-stop2`*`, ...)`

  where *`direction`* is the direction of the gradient and *`color-stop1`*, *`color-stop2`*, and so on are the colors and their stopping positions within the gradient.
- To create a radial gradient, use the function

  `radial-gradient(`*`shape size`* `at` *`position`*`, `*`color-stop1`*`, `*`color-stop2`*`, ...)`

  where *`shape`* defines the shape of the gradient, *`size`* sets the gradient size, *`position`* places the center of the gradient, and *`color-stop1`*, *`color-stop2`*, and so on are the colors and their stopping positions within the gradient.
- To repeat a gradient, use the functions

  `repeating-linear-gradient(`*`params`*`)`
  `repeating-radial-gradient(`*`params`*`)`

  where *`params`* are the parameters of the `linear-gradient` or the `radial-gradient` functions.

The last visual effect that Kevin wants you to add to the Komatsu Family page is to make the figure box semi-transparent so that it blends in better with its background.

INSIGHT

Gradients as Images

Gradients can be treated as images and thus can be tiled within a background or used with the `border-image` style. For example, the following style rule creates a background of radial gradients repeated in the horizontal and vertical directions:

```
background-size: 50% 50%;
background-image-repeat: repeat;
background-image: radial-gradient(circle, yellow, blue);
```

Note that setting the background size to 50% 50% sets the width and height of each radial gradient to half of the width and height of the object. By setting the `background-image-repeat` style to `repeat`, the entire background is filled with radial gradients, resulting in two rows of two gradient images.

The following code shows how to use a linear gradient as a border image:

```
border: 30px solid transparent;
border-image: (linear-gradient(yellow, blue)), 15);
```

with the image file replaced by the linear-gradient() function. The gradient is generated for the entire object but is trimmed with a slice width of 15px to form the gradient border.

TRY IT

You can explore gradients as backgrounds in the demo_linear_image.html and demo_radial_image.html files in the html04 ▸ demo folder

TRY IT

You can explore gradient borders using the demo_gradient_border.html file in the html04 ▸ demo folder.

Creating Semi-Transparent Objects

In Tutorial 2, you learned that you could create semi-transparent colors that blend with the background color. You can also create whole page objects that are semi-transparent using the following `opacity` property:

```
opacity: value;
```

where *value* ranges from 0 (completely transparent) up to 1 (completely opaque). For example, the following style rule makes the page body 70% opaque, allowing a bit of the browser window background to filter through

```
body {
   opacity: 0.7;
}
```

REFERENCE

Making a Semi-transparent Object

- To make a page object semi-transparent, use the property

  ```
  opacity: value;
  ```

  where *value* ranges from 0 (completely transparent) up to 1 (completely opaque).

Kevin suggests that you set the opacity of the figure box to 55% in order to blend the figure box with the paper texture background you added to the `article` element.

To create a semi-transparent object:

1. Return to the **tb_visual1.css** file in your editor and scroll up to the Figure Box Styles section.
2. Within the style rule for the `figure` element, insert the following style:

```
opacity: 0.55;
```

Figure 4–43 highlights the code to make the figure box semi-transparent.

Figure 4–43 Creating a semi-transparent object

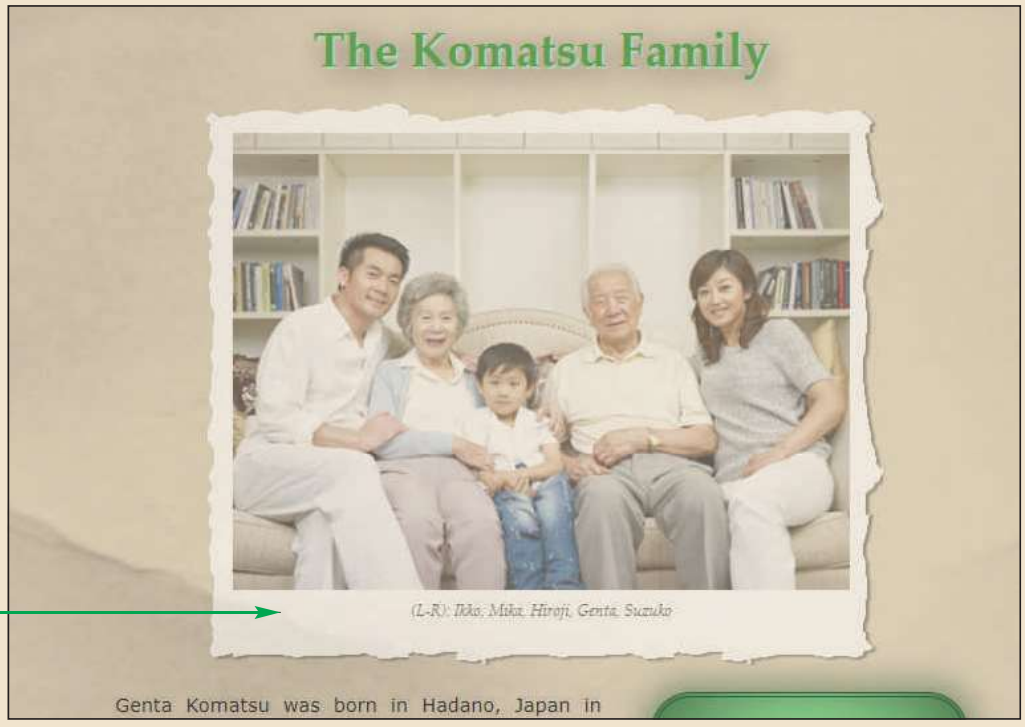
sets the opacity of the figure box to 55%

```
figure {
   border-style: solid;
   border-width: 25px;
   border-image: url(tb_border.png) 50 repeat;
   margin: 20px auto 0px;
   opacity: 0.55;
   width: 80%;
}
```

3. Save your changes and reload tb_komatsu.html in your browser. Figure 4–44 displays the semi-transparent figure box with part of the background paper texture showing through.

Figure 4–44 Changing the opacity of the figure box

part of the background page texture shows through in the figure box



PROSKILLS

Written Communication: How to Use Visual Effects

The CSS visual styles can add striking effects to your website, but they might not be supported by older browsers. This leaves you with the dilemma of when and how to use these styles. Here are some tips to keep in mind when applying visual effects to your website:

- Because not every user will be able to see a particular visual effect, design your page so that it is still readable to users with or without the effect.
- Be aware that some visual effects that flicker or produce strobe-like effects can cause discomfort and even photo-epileptic seizures in susceptible individuals. Avoid clashing color combinations and optical illusions that can cause these conditions.
- If you need to create a cross-browser solution, use browser extensions and be aware that the browser extension syntax might not match the syntax of the CSS standard.
- Consider using graphic images to create your visual effects. For example, rather than using the CSS gradient functions, create a background image file containing the gradient effect of your choice.

No matter how you employ visual effects on your website, remember that the most important part of your site is its content. Do not let visual effects distract from your content and message.

At this point you've completed your work on the design of the Komatsu Family page. In the next session, you will learn how to use CSS to apply transformations and filters. You will also learn how to work with image maps to create linkable images. Close any open files now.

REVIEW

Session 4.2 Quick Check

1. Provide a style rule to create a red text shadow that is 5 pixels to the right and 10 pixels up from the text with a blur of 15 pixels.
   a. `text-shadow: red 5px 10px 15px;`
   b. `text-shadow: red -5px 10px 15px;`
   c. `text-shadow: red 5px -10px 15px;`
   d. `text-shadow: red -5px -10px 15px;`
2. Provide a style rule to add a blue drop shadow to a page object that is 2 pixels to the left, 5 pixels up and with a blur radius of 8 pixels.
   a. `box-shadow: blue 2px 5px 8px;`
   b. `box-shadow: blue -2px 5px 8px;`
   c. `box-shadow: blue 2px -5px 8px;`
   d. `box-shadow: blue -2px -5px 8px;`
3. Provide a style to add a green interior drop shadow that is 2 pixels to the left, 5 pixels up and a blur radius of 8 pixels.
   a. `box-shadow: green 2px 5px 8px;`
   b. `box-shadow: green 2px 5px 8px inset;`
   c. `box-shadow: green -2px -5px 8px;`
   d. `box-shadow: green -2px -5px 8px inset;`

4. Provide a style rule to create a red halo effect with no shadow offset, a blur of 15 pixels and a shadow size that is 10 pixels larger than the element.
   a. `box-shadow: red 0px 0px 15px 10px;`
   b. `box-shadow: red 0px 0px 10px 15px;`
   c. `box-shadow: red 15px 10px 0px 0px;`
   d. `box-shadow: red 10px 15p x0px 0px;`
5. Provide code for a linear gradient that moves in the direction of the lower-left corner of the element through the colors: orange, yellow, and green.
   a. `linear-gradient(left bottom, orange, yellow, green)`
   b. `linear-gradient(bottom left, orange, yellow, green)`
   c. `linear-gradient(to left bottom, orange, yellow, green)`
   d. `linear-gradient(from right top, orange, yellow, green)`
6. Create a linear gradient that moves at a 15 degree angle with the color orange stopping at 10% of the background, yellow stopping at 50%, and green stopping at 55%.
   a. `linear-gradient(15deg, 10% orange, 50% yellow, 55% green)`
   b. `linear-gradient(15deg, orange 10%, yellow 50%, green 55%)`
   c. `linear-gradient(15deg, orange 10% yellow 50% green 55%)`
   d. `linear-gradient(195deg, 10% orange, 50% yellow, 55% green)`
7. Create a radial gradient that extends to the farthest background corner, going through the colors orange, yellow, and green.
   a. `radial(farthest-corner, orange, yellow, green)`
   b. `radial-gradient(from farthest-corner, orange, yellow, green)`
   c. `radial-gradient(farthest-corner, orange, yellow, green)`
   d. `radial(farthest-corner, orange, yellow, green)`
8. Create a repeating circular gradient of orange, yellow, and green bands centered at the right edge of the element with the colors stopped at 10%, 20%, and 30% respectively.
   a. `radial-gradient(circle at right center, orange 10%, yellow 20%, green 30%)`
   b. `radial-gradient-repeat(circle at right center, orange 10%, yellow 20%, green 30%)`
   c. `radial-gradient-repeat(circle at right center, orange 10% 10%, yellow 20% 20%, green 30% 30%)`
   d. `repeating-radial-gradient(circle at right center, orange 10%, yellow 20%, green 30%)`
9. Create a style rule to set the opacity to 75%.
   a. `transparency: 25%;`
   b. `transparency: 0.25;`
   c. `opacity: 75%;`
   d. `transform: opacity(0.75);`

Session 4.3 Visual Overview:

Perspective is used in 3D transformations to measure how rapidly objects appear to recede from or approach the viewer.

The **transform** property is used to rotate, rescale, skew, or shift a page object.

The **filter** property is used to modify an object's color, brightness, contrast, or general appearance.

The grayscale function displays the object in grayscale.

The saturate and contrast functions increase the color saturation by 50% and increase the color contrast by 20%.

```
/* Transformation and Filter Styles */

article {
   perspective: 600px;
}

figure#figure1 {
   transform: rotateX(30deg) translateZ(50px);

   filter: sepia(0.8);
}

figure#figure2 {
   transform: rotate(-40deg) scale(0.8, 0.8) translate(20px, -100px)
              rotateZ(30deg) rotateY(60deg);

   filter: grayscale(1);
}

figure#figure3 {
   transform: rotate(10deg) scale(0.9, 0.9)
              translateY(-120px) rotateY(-70deg) translateZ(-20px);

   filter: saturate(1.5) contrast(1.2);
}
```

The rotateX and translateY functions rotate the object 30° around the *x*-axis and move it 50 pixels toward the viewer.

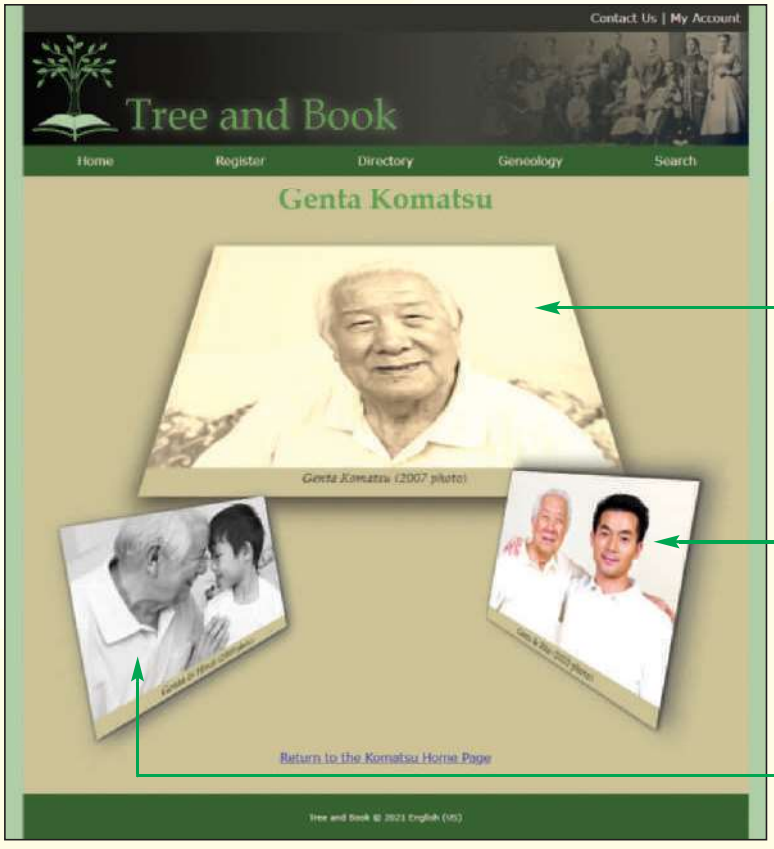
The sepia function displays the object in a sepia tone.

The rotateZ and rotateY functions rotate the object 30° around the *z*-axis and 60° around the *y*-axis.

The scale function reduces the object to 90% of its default size.

Logo Design Studio Pro; Source: wiki Media;

Transformations and Filters



Source: Design Studio Pro; Source: Wikimedia Commons; imtmphoto/Shutterstock.com

Transforming Page Objects

In this session, you will examine some CSS styles that can be used to transform the appearance of page objects through rotation, rescaling, and translation in space. To accomplish these transformations, you'll use the following `transform` property:

```
transform: effect(params);
```

where *effect* is a transformation function that will be applied to the page object and *params* are any parameters required by the function. Figure 4–45 describes some of the CSS transformation functions.

Figure 4–45 CSS 2D transformation functions

Function	Description
`translate(offX, offY)`	Moves the object *offX* pixels to the right and *offY* pixels down; negative values move the object to the left and up
`translateX(offX)`	Moves the object *offX* pixels to the right; negative values move the object to the left
`translateY(offY)`	Moves the object *offY* pixels down; negative values move the object up
`scale(x, y)`	Resizes the object by a factor of *x* horizontally and a factor of *y* vertically
`scaleX(x)`	Resizes the object by a factor of *x* horizontally
`scaleY(y)`	Resizes the object by a factor of *y* horizontally
`skew(angleX, angleY)`	Skews the object by *angleX* degrees horizontally and *angleY* degrees vertically
`skewX(angleX)`	Skews the object by *angleX* degrees horizontally
`skewY(angleY)`	Skews the object by *angleY* degrees vertically
`rotate(angle)`	Rotates the object by *angle* degrees clockwise; negative values rotate the object counter-clockwise
`matrix(n, n, n, n, n, n)`	Applies a 2D transformation based on a matrix of six values

For example, to rotate an object 30° clockwise, you would apply the following style using the `rotate` function:

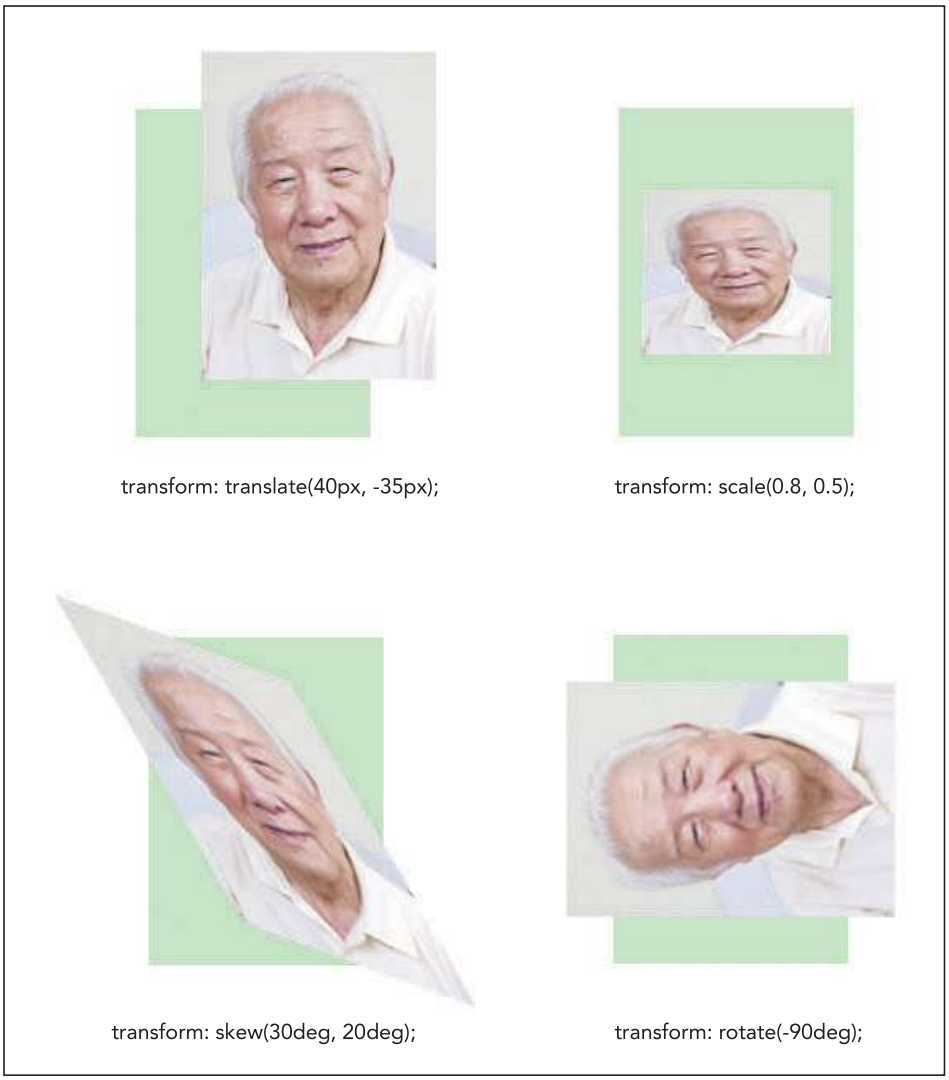
```
transform: rotate(30deg);
```

To rotate an object counter-clockwise, you would use a negative value for the angle of rotation. Thus, the following style rotates an object 60° counter-clockwise:

```
transform: rotate(-60deg);
```

Figure 4–46 displays the effects of other transformation functions on a sample page image.

Figure 4–46 Examples of CSS Transformations



TRY IT

You can explore different 2D CSS transformations using the demo pages demo_transform2d.html and demo_transform2dm.html from the html04 ▸ demo folder.

Transforming an object has no impact on the page layout. All of the other page objects will retain their original positions.

You can apply multiple transformations by placing the effect functions in a space-separated list. In this situation, transformations are applied in the order listed. For example, the following style first rotates the object 30° clockwise and then shifts it 20 pixels to the right.

```
transform: rotate(30deg) translateX(20px);
```

REFERENCE

Applying a CSS Transformation

- To apply a transformation to a page object, use the property

```
transform: effect(params);
```

where *effect* is a transformation function that will be applied to the page object and *params* are any parameters required by the function.

The website has pages with photos for each individual in the Komatsu family. Kevin wants you to work on transforming the photos on Genta Komatsu's page. Kevin has already created the page content and a layout and typographical style sheet but wants you to work on the style sheet containing the visual effects. Open the Genta Komatsu page now.

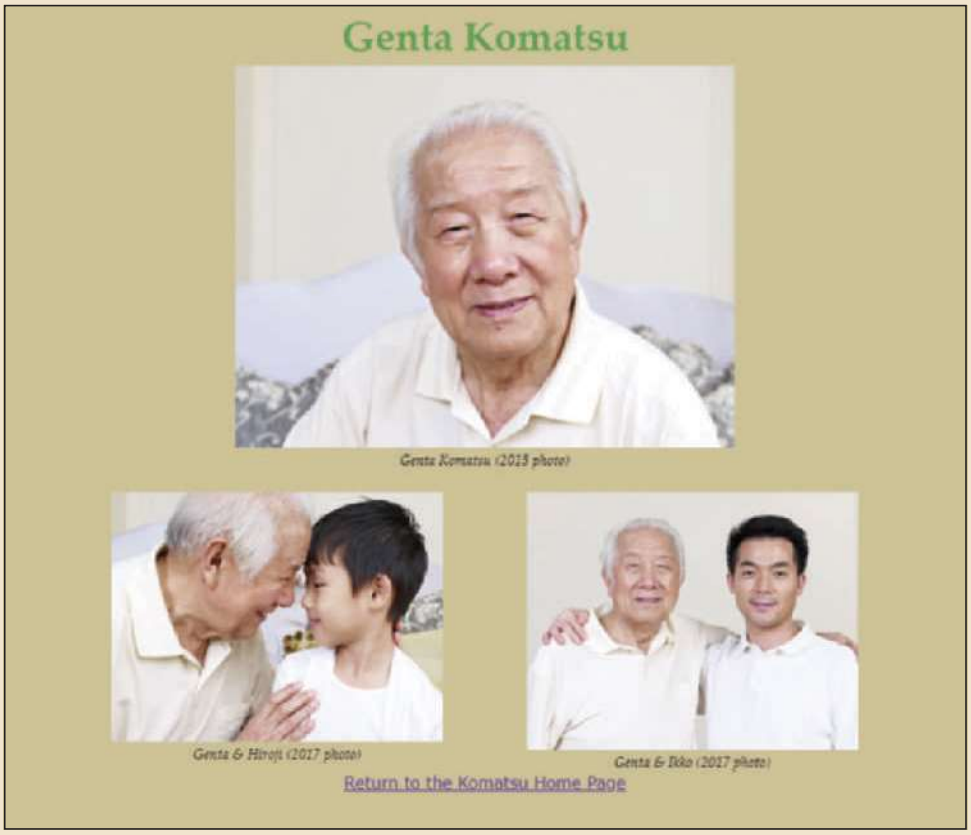
To open the Genta Komatsu page:

1. Use your editor to open the **tb_genta_txt.html** and **tb_visual2_txt.css** files from the html04 ▸ tutorial folder. Enter ***your name*** and ***the date*** in the comment section of both files and save them as **tb_genta.html** and **tb_visual2.css** respectively.
2. Return to the **tb_genta.html** file in your editor. Within the document head, insert the following `link` elements to link the page to the tb_reset.css, tb_styles2.css, and tb_visual2.css style sheet files.

```
<link href="tb_reset.css" rel="stylesheet" />
<link href="tb_styles2.css" rel="stylesheet" />
<link href="tb_visual2.css" rel="stylesheet" />
```

3. Take some time to scroll through the contents of the file. Note that the document content consists mainly of three figure boxes each containing a different photo of Genta Komatsu.
4. Close the file, saving your changes.
5. Open the **tb_genta.html** file in your browser. Figure 4–47 shows the initial layout and design of the page content.

Figure 4–47 Initial design of the Genta Komatsu page



Kevin feels that the page lacks visual interest. He suggests you transform the bottom row of photos by rotating them and shifting them upward to partially cover the main photo, creating a collage-style layout. Apply the `transform` property now to make these changes.

To apply the transform style:

1. Go to the **tb_visual2.css** file in your editor and scroll as needed to the Transformation Styles section.
2. Insert the following style rule to rotate the figure2 figure box 40° counter-clockwise, reduce it to 80% of its former size, and shift it 20 pixels to the right and 100 pixels up. Also, add a style to create a drop shadow using the code that follows:

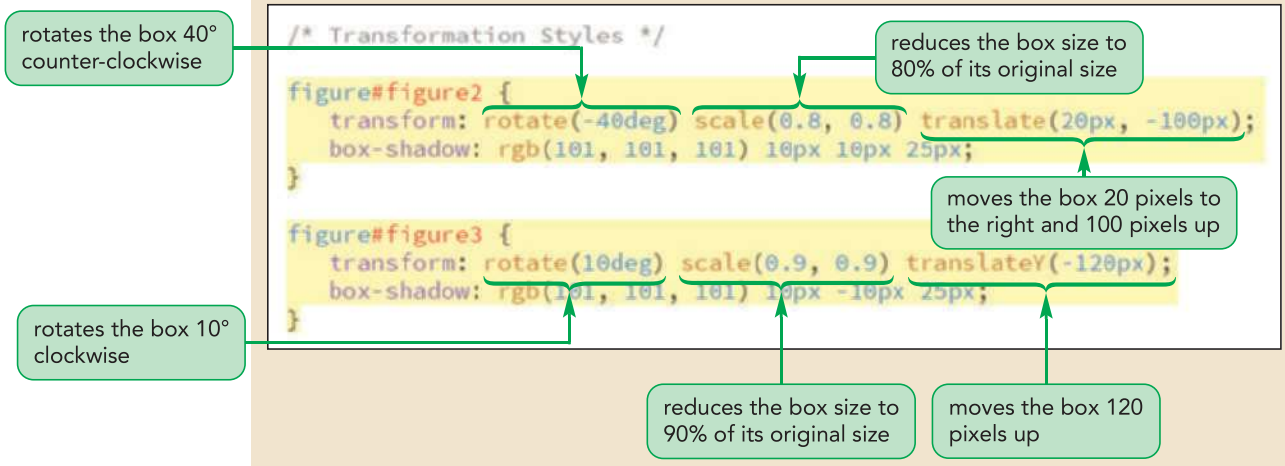
```
figure#figure2 {
   transform: rotate(-40deg) scale(0.8, 0.8)
              translate(20px, -100px);
   box-shadow: rgb(101, 101, 101) 10px 10px 25px;
}
```

3. Add the following style rule to rotate the figure3 figure box 10° clockwise, resize it to 90% of its current size, and shift it 120 pixels upward. Also add a drop shadow to the figure box using the following style rule:

```
figure#figure3 {
   transform: rotate(10deg) scale(0.9, 0.9)
              translateY(-120px);
   box-shadow: rgb(101, 101, 101) 10px -10px 25px;
}
```

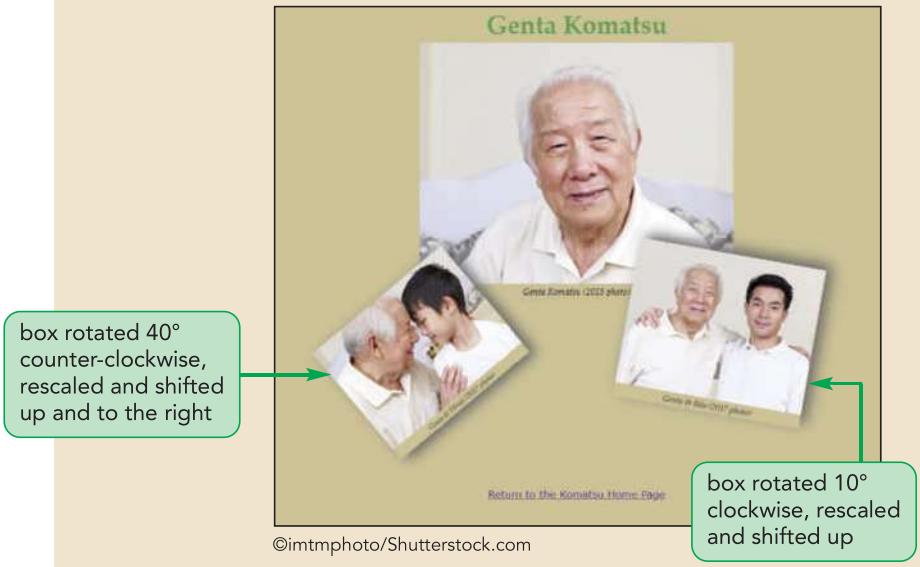
Figure 4–48 describes the newly added style rules.

Figure 4–48 Transforming the figure boxes



4. Save your changes to the file and then reload tb_genta.html in your browser. Figure 4–49 shows the revised design of the page's content.

Figure 4–49 Viewing the transformed figure boxes



©imtmphoto/Shutterstock.com

The transformations you applied rotated the figure boxes along a two-dimensional or 2D space that consisted of a horizontal and vertical axis. CSS also supports transformations that operate in a three-dimensional or 3D space.

INSIGHT

Setting the Transformation Origin

By default, transformations originate in the center of the page object. When an object is rotated, for example, it rotates the specified number of degrees around its horizontal and vertical center. If you wish to rotate around a different point, such as the object's left edge or bottom-right corner, you can modify the transformation origin using the following `transform-origin` property:

```
transform-origin: horizontal vertical;
```

where *horizontal* and *vertical* specify the location of the origin of the transformation. For example, the following set of style rules used in conjunction will rotate an object 30 degrees clockwise arounds its bottom-right corner:

```
transform: rotate(30deg);
transform-origin: right bottom;
```

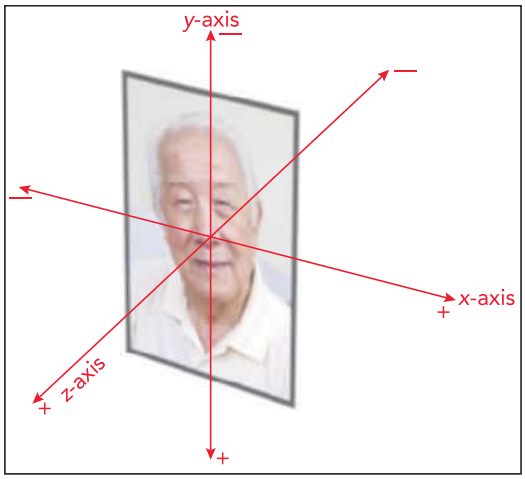
TRY IT

You can explore the `transform-origin` property in the demo_transform2d.html and demo_transform2dm.html files from the html04 ▸ demo folder.

Transformations in Three Dimensions

A **3D transformation** is a change that involves three spatial axes: an *x*-axis that runs horizontally across the page, a *y*-axis that runs vertically, and a *z*-axis that comes straight out of the page toward and away from the viewer. Positive values along the axes are to the right, down, and toward the reader; negative values are to the left, up, and away from the reader (see Figure 4–50.)

Figure 4–50 A page object viewed in 3D



With the addition of a third spatial axis, you can create effects in which an object appears to zoom toward and away from users, or to rotate in three dimensional space. Figure 4–51 describes the 3D transformations supported by CSS.

Figure 4–51 CSS 3D transformation functions

Function	Description
`translate3d(offX, offY, offZ)`	Shifts the object *offX* pixels horizontally, *offY* pixels vertically, and *offZ* pixels along the z-axis

`translateX(offX)`
`translateY(offY)`
`translateZ(offZ)`
	Shifts the object *offX*, *offY*, or *offZ* pixels along the specified axis

`rotate3d(x, y, z, angle)`	Rotates the object around the three-dimensional vector (*x*, *y*, *z*) at a direction of *angle*

`rotateX(angle)`
`rotateY(angle)`
`rotateZ(angle)`
	Rotates the object around the specified axis at a direction of *angle*

`scale3d(x, y, z)`	Resizes the object by a factor of *x* horizontally, a factor of *y* vertically, and a factor of *z* along the z-axis

`scaleX(x)`
`scaleY(y)`
`scaleZ(z)`
	Resizes the object by a factor of *x*, *y*, or *z* along the specified axis

`perspective(p)`	Sets the size of the perspective effect to *p*
`matrix3d(n, n, …, n)`	Applies a 3D transformation based on a matrix of 16 values

For example the following style rotates the object 60° around the *x*-axis, making it appear as if the top of the object is farther from the viewer and the bottom is closer to the viewer.

```
transform: rotateX(60deg);
```

To truly create the illusion of 3D space however, you also need to set the perspective of that space.

TRY IT

View 3D rotations and perspective values with the demo_3dview.html file in the html04 ▸ demo folder.

Understanding Perspective

Perspective is a measure of how rapidly objects appear to recede from the viewer in a 3D space. You can think of perspective in terms of a pair of railroad tracks that appear to converge at a point, known as the **vanishing point**. A smaller perspective

value causes the tracks to converge over an apparently shorter distance while a larger perspective value causes the tracks to appear to go farther before converging.

You define the perspective of a 3D space using the `perspective` property

```
perspective: value;
```

where *value* is a positive value that measures the strength of the perspective effect with lower values resulting in more extreme distortion. For example, the following style rule sets the perspective of the space within the `div` element to 400 pixels.

```
div {
   perspective: 400px;
}
```

Any 3D transformations applied to children of that `div` element will assume a perspective value of 400 pixels. Perspective can also be set for individual transformations using the following `perspective` function:

```
transform: perspective(value);
```

Thus, the following style rule sets the perspective only for the figure1 figure box within the `div` element as the figure box is rotated 60° around the *x*-axis.

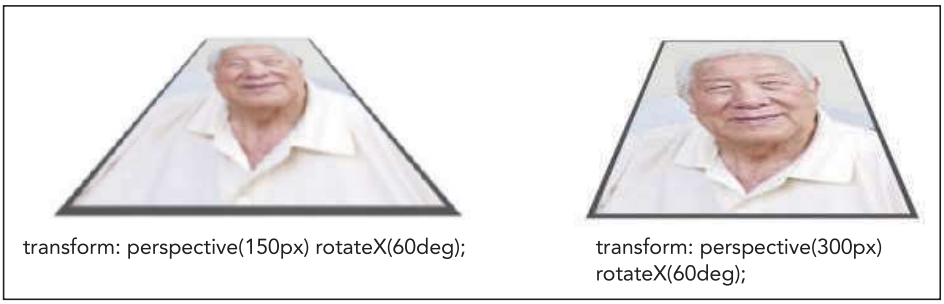
```
div figure#figure1 {
   transform: perspective(400px) rotateX(60deg);
}
```

You use the `perspective` property when you have several transformed objects within a container that all need to appear within the same 3D space with a common perspective. You use the `perspective` function when you have only one object that needs to be transformed in the 3D space. Figure 4–52 compares two different perspective values for an object rotated 60° around the *x*-axis in 3D space.

Figure 4–52 Transformations in three dimensions

TRY IT

Explore multiple 3D transformations with the demo_transform3dm.html file in the html04 ▸ demo folder.



transform: perspective(150px) rotateX(60deg);

transform: perspective(300px) rotateX(60deg);

Note that the smaller perspective value results in a more extreme distortion as the top of the object appears to more quickly recede from the viewer while the bottom appears to approach the viewer more rapidly.

REFERENCE

Setting Perspective in 3D

- To set the perspective for a container and the objects it contains, use the property

  ```
  perspective: value;
  ```

  where *value* is a positive value that measures the strength of the perspective effect with lower values resulting in more extreme distortion.
- To set the perspective of a single object or to set the perspective individually of objects within a group of objects, use the `perspective` function

  ```
  transform: perspective(value);
  ```

Add a 3D transformation to each of the three figure boxes in the Genta Komatsu page, making it appear that they have been rotated in three dimensional space along the *x*-, *y*-, and *z*-axes, setting the perspective value to 600 pixels for all of the objects in the page article.

To apply the 3D transformations:

1. Return to the **tb_visual2.css** file in your editor.
2. Directly after the Transformation Styles comment, insert the following style rule to set the perspective of the 3D space of the `article` element.

```
article {
   perspective: 600px;
}
```

3. Next, insert the following style rule for the figure1 figure box to rotate it 30° around the *x*-axis, shift it 50 pixels along the *z*-axis, and add a drop shadow.

```
figure#figure1 {
   transform: rotateX(30deg) translateZ(50px);
   box-shadow: rgb(51, 51, 51) 0px 10px 25px;
}
```

4. Add the following functions to the `transform` property for the figure2 figure box to rotate the box 30° around the *z*-axis and 60° around the *y*-axis:

```
rotateZ(30deg) rotateY(60deg)
```

5. Add the following functions to the `transform` property for the figure3 figure box to rotate the box counter-clockwise 70° around the *y*-axis and shift it 20 pixels away from the user along the *z*-axis:

```
rotateY(-70deg) translateZ(-20px)
```

Figure 4–53 highlights the 3D transformations styles in the style sheet.

Figure 4–53 Applying 3D transformations

sets the perspective of the article space to 600 pixels

rotates the box 30° around the x-axis and shifts it forward 50 pixels along the *z*-axis

rotates the box 30° around the *z*-axis and 60° around the *y*-axis

rotates the box 70° counter-clockwise around the *y*-axis and shifts it backward 20 pixels along the *z*-axis

adds a box shadow on the box's bottom border

```
/* Transformation Styles */

article {
   perspective: 600px;
}

figure#figure1 {
   transform: rotateX(30deg) translateZ(50px);
   box-shadow: rgb(51, 51, 51) 0px 10px 25px;
}

figure#figure2 {
   transform: rotate(-40deg) scale(0.8, 0.8)
              translate(20px, -100px)
              rotateZ(30deg) rotateY(60deg);
   box-shadow: rgb(101, 101, 101) 10px 10px 25px;
}

figure#figure3 {
   transform: rotate(10deg) scale(0.9, 0.9)
              translateY(-120px)
              rotateY(-70deg) translateZ(-20px);
   box-shadow: rgb(101, 101, 101) 10px -10px 25px;
}
```

6. Save your changes to the file and then reload tb_genta.html in your browser. Figure 4–54 shows the result of applying 3D transformations to each of the figure boxes on the page.

Figure 4–54 **Figure boxes in 3D space**

image rotated 30° around the *x*-axis and moved forward 50 pixels along the *z*-axis

image rotated 30° around the *z*-axis and 60° around the *y*-axis

image rotated 70° counter-clockwise around the *y*-axis and shifted backward 20 pixels along the *z*-axis

You have only scratched the surface of what can be done using transformations. For example, you can create a mirror image of an object by rotating it 180° around the *y*-axis. You can create virtual 3D objects like cubes that can be viewed from any angle or spun. You are only limited by your imagination.

INSIGHT

Managing a 3D Space

You might want to have several objects that coexist within the same 3D space. You can do this by creating a container for all those objects, allowing them to share a common 3D perspective using the following `transform-style` property:

```
transform-style: type;
```

where *type* is either `preserve-3d` to preserve the 3D space for all nested elements or `flat` to allow the nested elements to exist within their own separate 3D spaces. For example, the following style rules will pass the same 3D space to all elements nested within the container, including any value assigned to the `perspective` property.

```
#container {
   transform-style: preserve-3d;
}
```

An object in a 3D space is considered to have a front and a back. The default behavior is to allow any text or images on the front to "bleed through" to the back (thus appearing in reverse when the object is rotated around the x or y axes.) You can turn off this feature setting the `backface-visibility` property to `hidden`, which prevents text and images on the front face of the object from appearing on the back face. Setting `backface-visibility` to `visible` restores the default.

TRY IT

Explore managing multiple objects within a 3D space in the demo_preserve3d.html, demo_cards.html, and demo_cube.html files from the html04 ▶ demo folder.

Exploring CSS Filters

A final way to alter an object is through a CSS filter. Filters adjust how the browser renders an image, a background, or a border by modifying the object's color, brightness, contrast, or general appearance. For example, a filter can be used to change a color image to grayscale, increase the image's color saturation, or add a blurring effect. Filters are applied using the `filter` property

```
filter: effect(params);
```

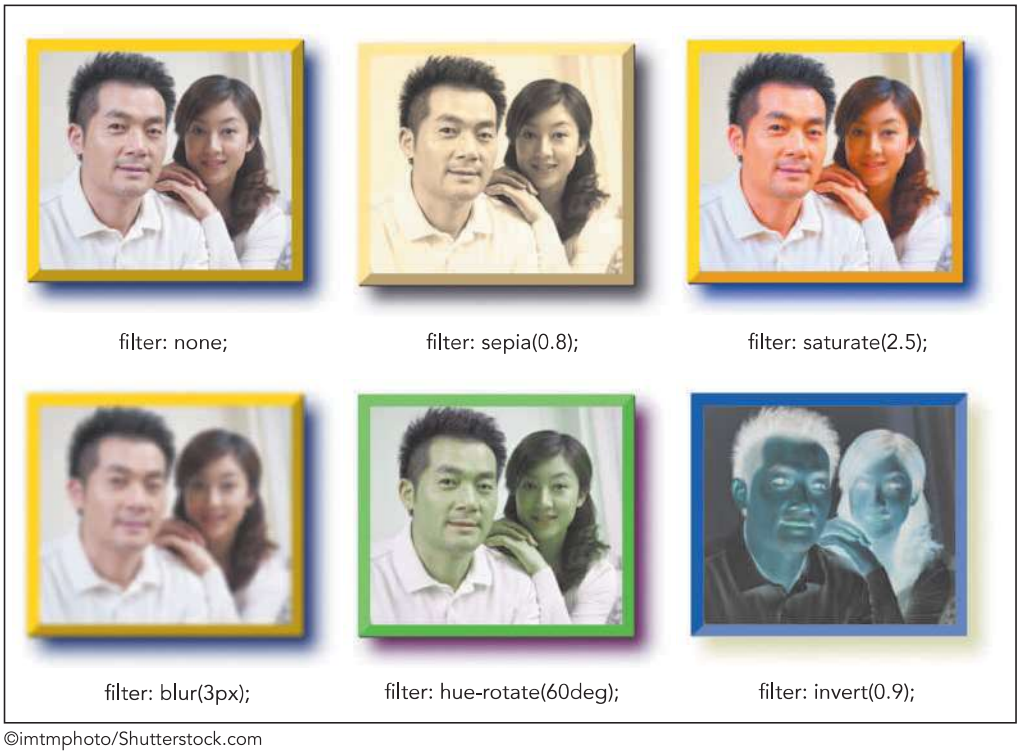
where *effect* is a filter function and *params* are the parameters of the function. Figure 4–55 describes the different filter functions supported by most current browsers.

Figure 4–55 CSS filter functions

Function	Description
`blur(length)`	Applies a blur to the image where *length* defines the size of blur in pixels
`brightness(value)`	Adjusts the brightness where values from 0 to 1 decrease the brightness and values greater than 1 increase the brightness
`contrast(value)`	Adjusts the contrast where values from 0 to 1 decrease the contrast and values greater than 1 increase the contrast
`drop-shadow(offsetX offsetY blur color)`	Adds a drop shadow to the image where *offsetX* and *offsetY* are horizontal and vertical distances of the shadow, *blur* is the shadow blurring, and *color* is the shadow color
`grayscale(value)`	Displays the image in grayscale from 0, leaving the image unchanged, up to 1, displaying the image in complete grayscale
`hue-rotate(angle)`	Adjusts the hue by *angle* in the color wheel where 0deg leaves the hue unchanged, 180deg displays the complimentary colors and 360deg again leaves the hue unchanged
`invert(value)`	Inverts the color from 0 (leaving the image unchanged), up to 1 (completely inverting the colors)
`opacity(value)`	Applies transparency to the image from 0 (making the image transparent), up to 1 (leaving the image opaque)
`saturate(value)`	Adjusts the color saturation where values from 0 to 1 decrease the saturation and values greater than 1 increase the saturation
`sepia(value)`	Displays the color in a sepia tone from 0 (leaving the image unchanged), up to 1 (image completely in sepia)
`url(url)`	Loads an SVG filter file from *url*

Figure 4–56 shows the impact of some of the filter functions on a sample image.

Figure 4–56 CSS filter examples



TRY IT

Explore the CSS filter styles with the demo_filter.html file in the html04 ▸ demo folder.

Filter functions can be combined in a space-separated list to create new effects. For example, the following style reduces the object's color contrast and applies a sepia tone.

```
filter: contrast(75%) sepia(100%);
```

With multiple filter effects, the effects are applied in the order they are listed. Thus, a style in which the sepia effect is applied first followed by the contrast effect will result in a different image than if the order is reversed.

REFERENCE

Applying a CSS Filter

- To apply a CSS filter to a page object, use the property

```
filter: effect(params);
```

where *effect* is a filter function and *params* are the parameters of the function.

Kevin wants you to apply filters to the photos in the Genta Komatsu page. He wants a sepia tone applied to the first photo, a grayscale filter applied to the second photo, and a color enhancement applied to the third photo.

To apply the CSS filters:

1. Return the **tb_visual2.css** file in your editor and go down to the Filter Styles section.
2. Change the figure1 figure box to a sepia tone by adding the following style rule:

```
figure#figure1 {
   filter: sepia(0.8);
}
```

3. Change the figure2 figure box to grayscale by adding the style rule:

```
figure#figure2  {
   filter: grayscale(1);
}
```

4. Increase the saturation and contrast for the figure3 figure box with the style rule:

```
figure#figure3  {
   filter: saturate(1.5) contrast(1.2);
}
```

Figure 4–57 highlights the CSS filters added to the style sheet.

Figure 4–57 Applying the filter property

```
/* Filter Styles */

figure#figure1 {
   filter: sepia(0.8);
}

figure#figure2 {
   filter: grayscale(1);
}

figure#figure3 {
   filter: saturate(1.5) contrast(1.2);
}
```

provides more cross-browser support by adding the WebKit browser extension

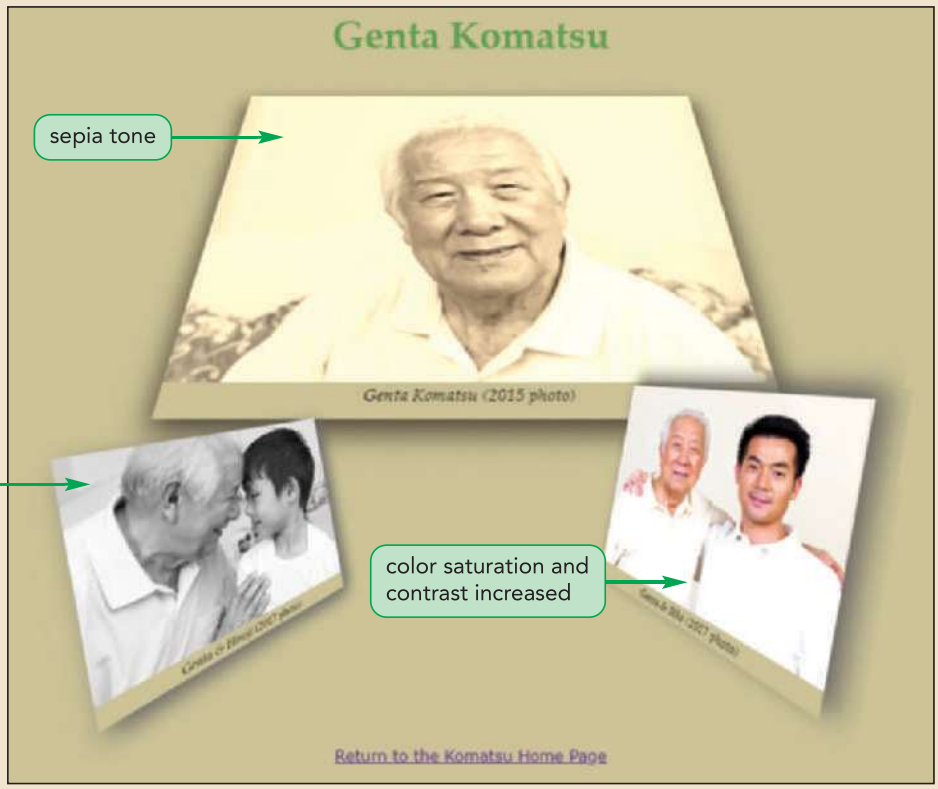
displays the figure1 figure box in sepia

displays the figure2 figure box in grayscale

increases the color saturation and contrast in the figure3 figure box

5. Save your changes to the file and then reload tb_genta.html in your browser. Figure 4–58 shows the final design of the Genta Komatsu page.

Figure 4–58 Filters applied to the web page photos



INSIGHT

Box Shadows and Drop Shadows

You may wonder why you need a `drop-shadow` filter if you already have the `box-shadow` property. While they both can be used to add shadowing to a page object, one important difference is that the `drop-shadow` filter creates a shadow that traces the shape of the object, while the `box-shadow` property always applies a rectangular shadow. Another important difference is that you can only change the size of a shadow using the `box-shadow` property. Thus, if you want to apply a drop shadow around objects such as text or a circular shape, use the `drop-shadow` filter. However, if you need to create an internal shadow or change the size of the drop shadow shadow, use the `box-shadow` property.

You've completed your redesign of the Genta Komatsu page by adding transformation and filter effects to make a more visually striking page. Kevin now wants to return to the page for the Komatsu family. He wants you to edit the family portrait on the page so that individual pages like the Genta Komatsu page can be accessed by clicking the person's face on the family portrait. You can create this effect using an image map.

Working with Image Maps

When you mark an inline image as a hyperlink, the entire image is linked to the same file; however, HTML also allows you to divide an image into different zones, or **hotspots**, which can then be linked to different URLs through information provided in an **image map**. HTML supports two kinds of image maps: client-side image maps and server-side image maps. A **client-side image map** is an image map that is defined within the web page and handled entirely by the web browser, while a **server-side image map** relies on a program running on the web server to create and administer the map. Generally client-side maps are easier to create and do not rely on a connection to the server in order to run.

Defining a Client-Side Image Map

Client-side image maps are defined with the following `map` element

```
<map name="text">
   hotspots
</map>
```

where *text* is the name of the image map and *hotspots* are defined regions within an image that are linked to different URLs. Client-side image maps can be placed anywhere within the body of a web page because they are not actually displayed by browsers but are simply used as references for mapping the locations of the hotspots within the image. The most common practice is to place a `map` element below the corresponding inline image.

Each hotspot within the `map` element is defined using the following `area` element:

```
<area shape="shape" coords="coordinates"
      href="url" alt="text" />
```

where *shape* is the shape of the hotspot region, *coordinates* are the list of points that define the boundaries of that region, *url* is the URL of the hypertext link, and *text* is alternate text displayed for non-graphical browsers.

TIP

Do not overlap the hotspots to avoid confusing the user about which hotspot is associated with which URL.

Hotspots can be created as rectangles, circles, or polygons (multisided figures) using *shape* values of `rect`, `circle`, and `poly` respectively. A fourth possible *shape* value, `default`, represents the remaining area of the inline image not covered by any hotspots. There is no limit to the number of hotspots you can add to an image map.

For rectangular hotspots, the `shape` and `coords` attributes have the general form:

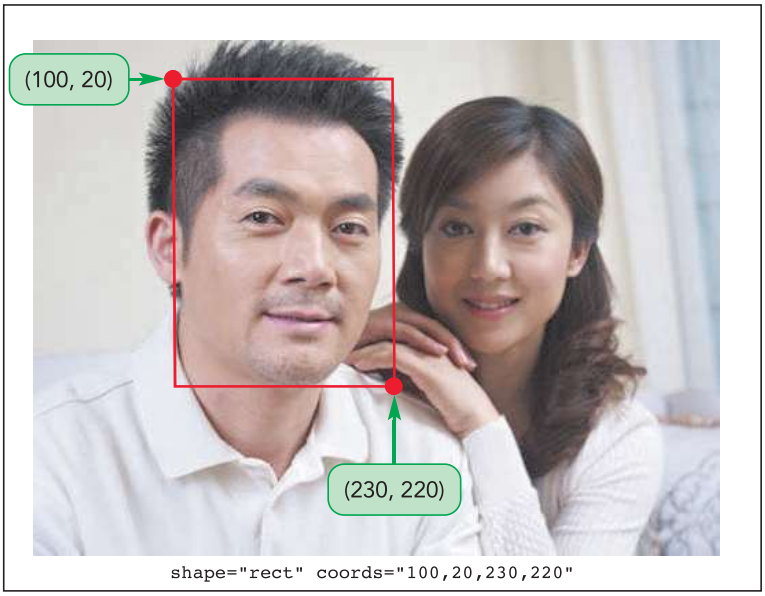
```
shape="rect" coords="left,top,right,bottom"
```

where *left*, *top* are the coordinates of the top-left corner of the rectangle and *right*, *bottom* are the coordinates of the bottom-right corner. Coordinates for hotspot shapes are measured in pixels and thus, the following attributes define a rectangular hotspot with the left-top corner at the coordinates (100, 20) and the right-bottom corner at (230, 220):

```
shape="rect" coords="100,20,230,220"
```

To determine the coordinates of a hotspot, you can use either a graphics program such as Adobe Photoshop or image map software that automatically generates the HTML code for the hotspots you define. Note that coordinates are always expressed relative to the top-left corner of the image, regardless of the position of the image on the page. For example, in Figure 4–59, the top-left corner of this rectangular hotspot is 100 pixels right of the image's left border and 20 pixels down from the top border.

Figure 4–59 Defining a rectangular hotspot

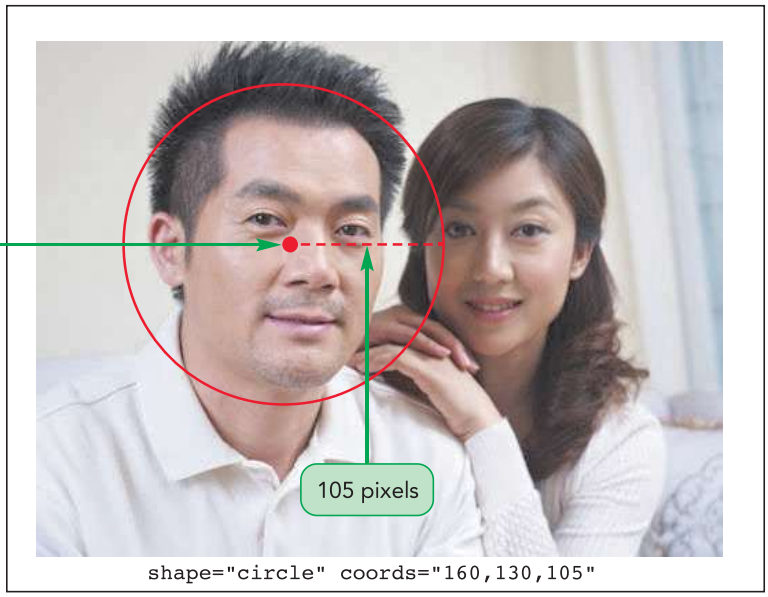


Circular hotspots are defined using the attributes

```
shape="circle" coords="x,y,radius"
```

where *x* and *y* are the coordinates of the center of the circle and *radius* is the circle's radius. Figure 4–60 shows the coordinates for a circular hotspot where the center of the circle is located at the coordinates (160, 130) with a radius of 105 pixels.

Figure 4–60 Defining a circular hotspot

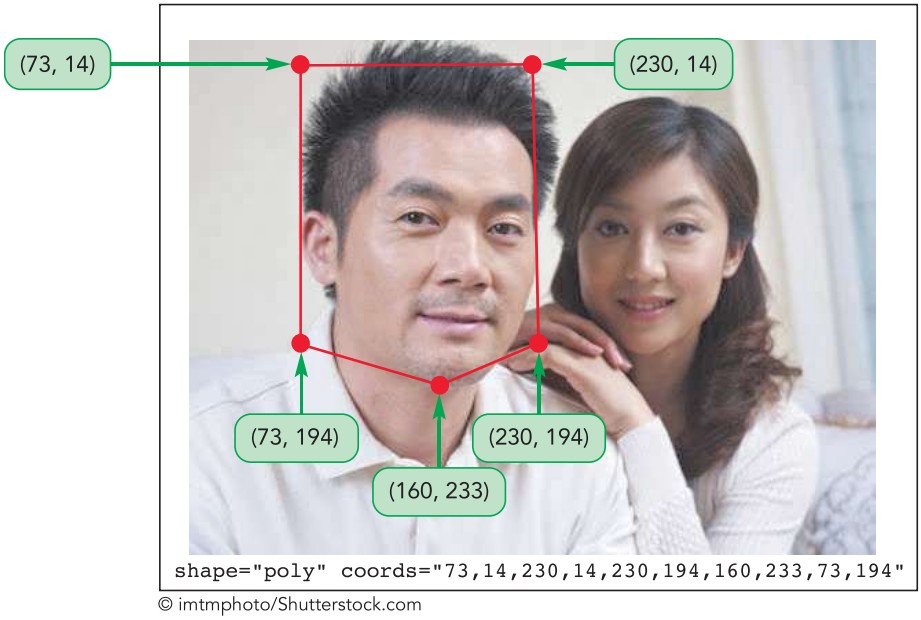


Polygonal hotspots have the attributes

```
shape="poly" coords="x1,y1,x2,y2,…"
```

where (*x1*, *y1*), (*x2*, *y2*), … set the coordinates of each vertex in the shape. Figure 4–61 shows the coordinates for a 5-sided polygon.

Figure 4–61 Defining a polygonal hotspot



TIP

Default hotspots should always be listed last.

To define the default hotspot for an image, create the following hotspot:

```
shape="default" coords="0,0,width,height"
```

where *width* is the width of the image in pixels and *height* is the image's height. Any region in the image that is not covered by another hotspot activates the default hotspot link.

REFERENCE

Creating an Image Map

- To create an image map, use

```
<map name="text">
   hotspots
</map>
```

where *text* is the name of the image map and *hotspots* are the hotspots within the image.

- To define each hotspot, use

```
<area shape="shape" coords="coordinates" href="url" alt="text" />
```

where *shape* is the shape of the hotspot region, *coordinates* list the points defining the boundaries of the region, *url* is the URL of the hypertext link, and *text* is alternate text that is displayed for non-graphical browsers.

- To define a rectangular hotspot, use the `shape` and `attribute` values

```
shape="rect" coords="left,top,right,bottom"
```

where *left*, *top* are the coordinates of the top-left corner of the rectangle and *right*, *bottom* are the coordinates of the bottom-right corner.

- To define a circular hotspot, use

```
shape="circle" coords="x,y,radius"
```

where *x* and *y* are the coordinates of the center of the circle and *radius* is the circle's radius.

- To define a polygonal hotspot, use

```
shape="poly" coords="x1,y1,x2,y2,..."
```

where (*x1*, *y1*), (*x2*, *y2*), and so on provide the coordinates of each vertex in the multisided shape.

- To define the default hotspot link, use

```
shape="default" coords="0,0,width,height"
```

where *width* and *height* is the width and height of the image.

Kevin has provided you with the coordinates for five rectangular hotspots to cover the five faces on the Komatsu family portrait. Add an image map named "family_map" to the tb_komatsu.html page with rectangular hotspots for each of the faces in the family portrait.

To create an image map:

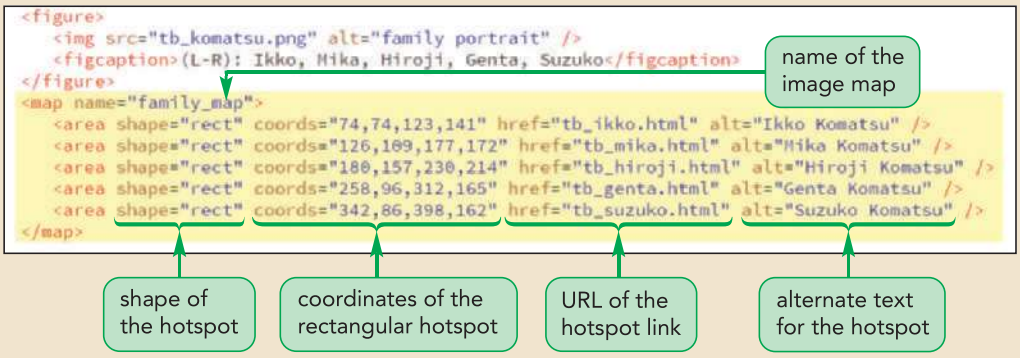
1. Open or return to the **tb_komatsu.html** file in your editor.
2. Directly below the figure box, insert the following HTML code:

```
<map name="family_map">
   <area shape="rect" coords="74,74,123,141"
    href="tb_ikko.html" alt="Ikko Komatsu" />
   <area shape="rect" coords="126,109,177,172"
    href="tb_mika.html" alt="Mika Komatsu" />
   <area shape="rect" coords="180,157,230,214"
    href="tb_hiroji.html" alt="Hiroji Komatsu" />
```

```
    <area shape="rect" coords="258,96,312,165"
     href="tb_genta.html" alt="Genta Komatsu" />
    <area shape="rect" coords="342,86,398,162"
     href="tb_suzuko.html" alt="Suzuko Komatsu" />
</map>
```

Figure 4–62 highlights the HTML code for the image map and hotspots.

Figure 4–62 Inserting an image map



3. Save your changes to the file.

With the image map defined, your next task is to apply that map to the image in the figure box.

Applying an Image Map

To apply an image map to an image, you add the following `usemap` attribute to the `img` element

```
<img src="url" alt="text" usemap="#map" />
```

where *map* is the name assigned to the image map within the current HTML file.

REFERENCE

Applying an Image Map

- To apply an image map to an image, add the `usemap` attribute to the `img` element

  ```
  <img src="url" alt="text" usemap="#map" />
  ```

  where *map* is the name assigned to the image map.

Apply the family_map image map to the figure box and then test it in your web browser.

To apply an image map:

1. Add the attribute **usemap="#family_map"** to the `img` element for the family portrait.

   Figure 4–63 highlights the code to apply the image map.

Figure 4–63 Applying an image map

Applies the family_map image map to the image

```
<figure>
   <img src="tb_komatsu.png" alt="family portrait" usemap="#family_map" />
   <figcaption>(L-R): Ikko, Mika, Hiroji, Genta, Suzuko</figcaption>
</figure>
```

2. Save your changes to the file and then reload tb_komatsu.html in your browser.
3. Click the five faces in the family portrait and verify each face is linked to a separate HTML file devoted to that individual. Use the link under the image of each individual to return to the home page.

Kevin likes the addition of the image map and plans to use it on other photos in the website.

PROSKILLS

Problem Solving: Image Maps with Flexible Layouts

Image maps are not easily applied to flexible layouts in which the size of the image can change based on the size of the browser window. The problem is that, because hotspot coordinates are expressed in pixels, they don't resize and will not point to the correct region of the image if the image is resized.

One way to deal with flexible layouts is to create hotspots using hypertext links that are sized and positioned using relative units. The image and the hypertext links would then be nested within a `figure` element as follows:

```
<figure class="map">
   <img src="image" alt="" />
   <a href="url" id="hotspot1"></a>
   <a href="url" id="hotspot2"></a>
   …
</figure>
```

The figure box itself needs to be placed using relative or absolute positioning and the image should occupy the entire figure box. Each hypertext link should be displayed as a block with width and height defined using percentages instead of pixels and positioned absolutely within the figure box, also using percentages for the coordinates. As the figure box is resized under the flexible layout, the hotspots marked with the hypertext links will automatically be resized and moved to match. The opacity of the hotspot links should be set to 0 so that the links do not obscure the underlying image file. Even though the hotspots will be transparent to the user, they will still act as hypertext links.

This approach is limited to rectangular hotspots. To create a flexible layout for other shapes, you need to use a third-party add-in that automatically resizes the shape based on the current size of the image.

You've completed your work on the Komatsu Family pages for *Tree and Book*. Kevin will incorporate your work and ideas with other family pages as he continues on the site redesign. He'll get back to you with more projects in the future. For now you can close any open files or applications.

REVIEW

Session 4.3 Quick Check

1. Provide the transformation to shift a page object 5 pixels to the right and 10 pixels up.
   a. `transform: translate(5px, 10px);`
   b. `transform: translate(5px, -10px);`
   c. `transform: translate(-5px, -10px);`
   d. `translate: -5px 10px;`
2. Provide the transformation to reduce the horizontal and vertical size of an object by 50%.
   a. `scale: 0.5, 0.5;`
   b. `transform: scale(0.5, 0.5);`
   c. `transform: rescale(0.5, 0.5);`
   d. `transform: resize(0.5, 0.5);`
3. Provide the transformation to rotate an object 30° counter-clockwise around the x-axis.
   a. `transform: rotate(-30deg);`
   b. `transform: rotate(30deg);`
   c. `transform: rotateX(30deg);`
   d. `transform: rotateX(-30deg);`
4. Provide the filter to increase the brightness of an object by 20%.
   a. `filter: brightness(20%);`
   b. `filter: brightness(0.2);`
   c. `filter: brightness(1.2);`
   d. `brightness: 0.2;`
5. Provide the filter to decrease the contrast of an object by 30%.
   a. `filter: contrast(0.3);`
   b. `filter: contrast(0.7);`
   c. `filter: contrast(30%);`
   d. `filter: contrast(-0.3);`
6. Provide the code to create a triangular hotspot with vertices at (200, 5), (300, 125), and (100, 125), linked to the info.html file.
   a. `<area type="poly" coords="200, 5, 300, 125, 100, 125" href="info.html" />`
   b. `<area type="triangle" coords="200, 5, 300, 125, 100, 125" href="info.html" />`
   c. `<area type="draw" coords="200, 5, 300, 125, 100, 125" href="info.html" />`
   d. `<area type="poly" coords="5, 200, 125, 300, 125, 100" href="info.html" />`
7. Provide code to attach the logo.png image to the mapsites image map:
   a. `<img src="logo.png" map="mapsites" alt="" />`
   b. `<img src="logo.png" map="#mapsites" alt="" />`
   c. `<img src="logo.png" imgmap="mapsites" alt="" />`
   d. `<img src="logo.png" usemap="#mapsites" alt="" />`
8. Provide a style rule to transfer 3D space from a container element to its nested elements.
   a. `transform-style: perspective;`
   b. `preserve-3d: true;`
   c. `transfer-style: preserve-3d;`
   d. `transform-style: preserve-3d;`