HTML5, CSS3, and JavaScript

6th Edition

Tutorial 4

Graphic Design with CSS

Carey

Objectives

- Create a figure box
- Add a background image
- Add a border to an element
- Create rounded borders
- Create a graphic border
- Create a text shadow

Objectives (continued)

- Create a box shadow
- Create linear and radial gradients
- Set the opacity of an element
- Apply a 2D and 3D transformation
- Apply a CSS filter
- Create an image map

Creating Figure Boxes

- `figure` and `figcaption` elements:
  - They are used by HTML5 to introduce structural elements
  - The `figcaption` element is optional
  - The `figcaption` element can be placed directly before or after a figure box content

Creating Figure Boxes (continued 1)

- The general syntax to create a figure box is as follows:

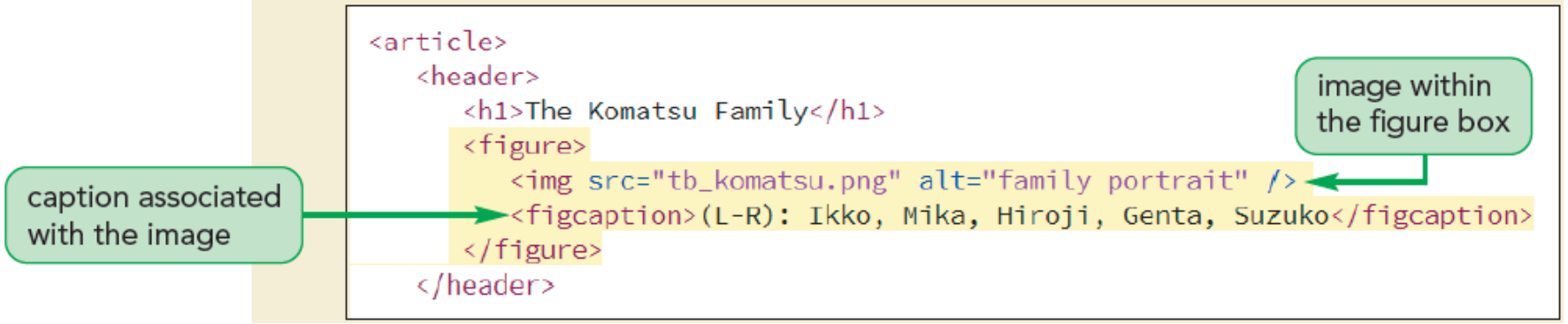
```
<figure>
content
<figcaption>caption text</figcaption>
</figure>
```

  - *content* is the content appearing in a figure box
  - *caption text* is the description text that accompanies the figure

Creating Figure Boxes (continued 2)

- The `figure` element – Used to mark a page content that should stand apart from the main content

Figure 4-1 Inserting a figure box



Exploring Background Styles

- The use of images for backgrounds is supported by CSS using the following `background-image` style:

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

  - *url* specifies the name and location of the background image

Tiling a Background Image

- **Tiling** – The process of repeating an image both vertically and horizontally until the background is filled
- The type of tiling can be specified by applying the following `background-repeat` style:

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

where *type* can be `repeat` (the default), `repeat-x`, `repeat-y`, `round`, or `space`

Tiling a Background Image (continued)

Figure 4-6 Tiled background image in the browser window

tiled image in browser window background

page body

Attaching the Background Image

- A background image is attached to its element so that it scrolls when the element content is scrolled
- The attachment can be changed using the following property:

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

where *type* is `scroll` (the default), `fixed`, or `local`

Attaching the Background Image (continued 1)

- *type* in the `background-attachment` property:
  - `scroll` sets the background to scroll with the element content
  - `fixed` creates a background that stays in place even as the element content is scrolled
  - `local` allows the element background to scroll along with the content within the box

Attaching the Background Image (continued 2)

- **Watermarks**:
  - Translucent graphics displayed behind a content
  - They can be created using fixed backgrounds
  - Often used to indicate that a content material is copyrighted

Setting the Background Image Position

- By default, background images are placed in an element's top-left corner
- The following property can be used to set the position of a background image:

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

where *horizontal* and *vertical* provide the coordinates of an image within the element background

Setting the Background Image Position (continued)

- Keywords to position a background image are as follows:
  - `left`, `center`, and `right` are used to position the background horizontally
  - `top`, `center`, and `bottom` are used to position the background vertically
- The `background-position` property is only useful for non-tiled images

Defining the Extent of the Background

- An element's background extends only through the padding space excluding the border space
- This can be changed using the following property:

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

Defining the Extent of the Background (continued)

- *type* in the `background-clip` property:
  - `content-box` extends the background only through the element content
  - `padding-box` extends the background through the padding space
  - `border-box` extends the background through the border space

Sizing and Clipping an Image

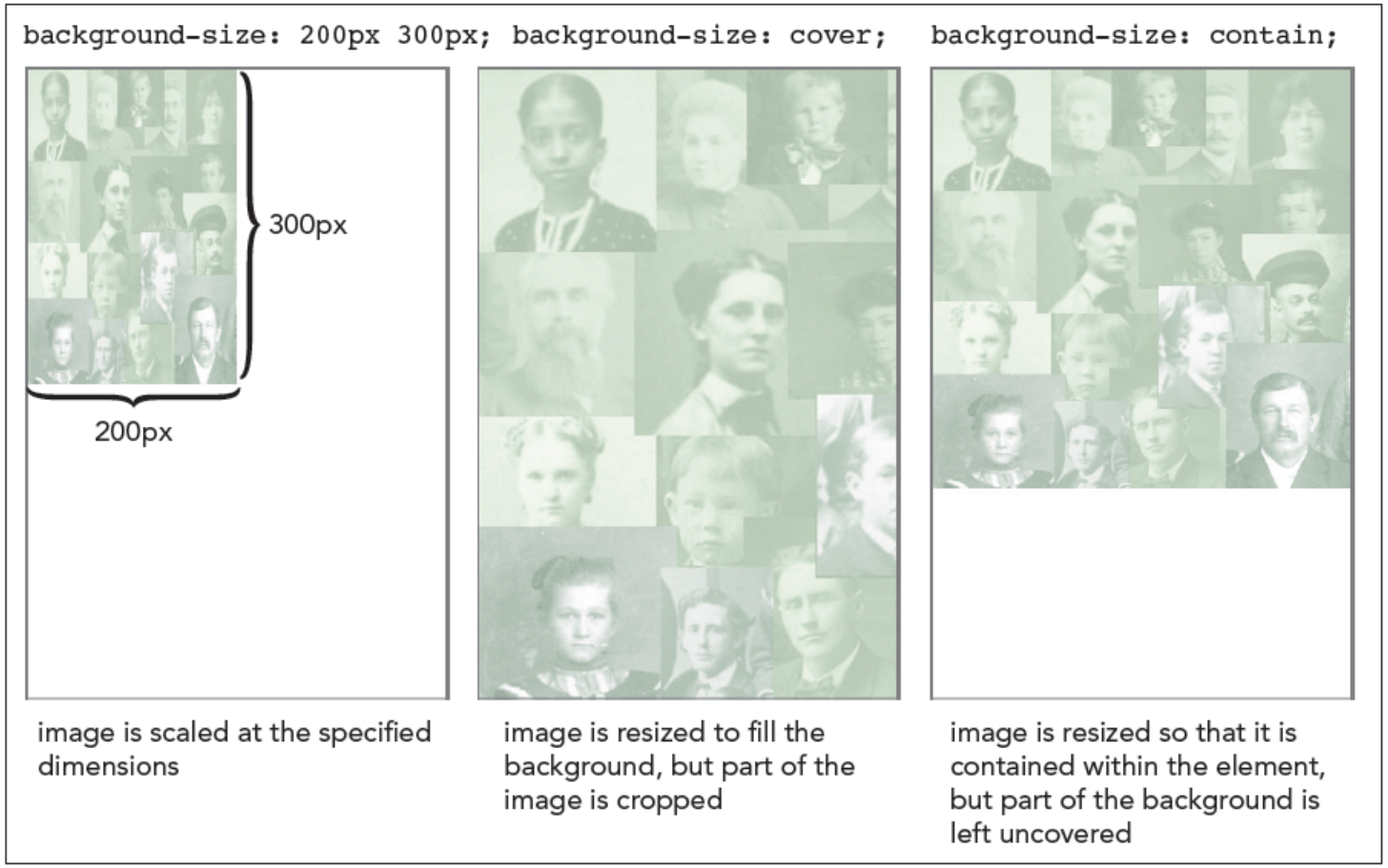
- By default, the size of a background image equals the size stored in its image file
- This size can be changed by using the following property:

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

where *width* and *height* can be specified in pixels or by using the keywords `auto`, `cover`, and `contain`

Sizing and Clipping an Image (continued)

Figure 4-7 Examples of background-size types



The background Property

- Different background options can be organized using the following property:

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

  - *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

The background Property (continued)

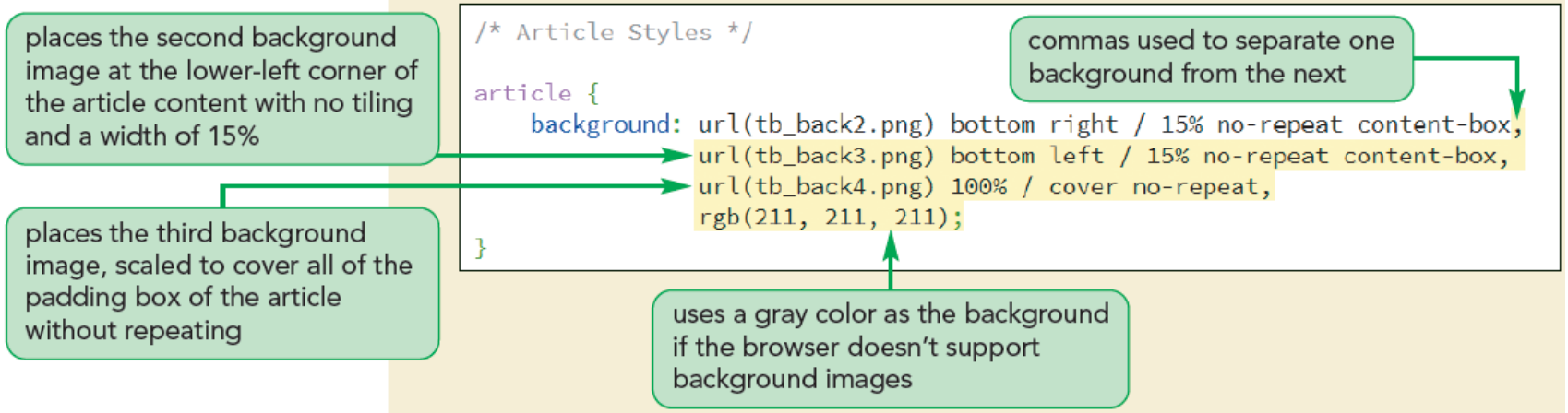
- *attachment* specifies whether the image scrolls with the content or is fixed
- *origin* defines how positions are measured on the background
- *clip* specifies the extent over which the background is spread

Adding Multiple Backgrounds

- Multiple backgrounds can be added to a single element by listing the backgrounds in the following comma-separated list:

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

Figure 4-10 Adding multiple background images



Setting Border Width and Color

- There are several style properties that can be used to format the border around each element using CSS
- To define the thickness of a specific border, the following property can be used:

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

  - *side* is either `top`, `right`, `bottom`, or `left`
  - *width* is the width of the border

Setting the Border Design

- The appearance of borders can be further defined by using the following style:

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

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

adds a 1-pixel solid gray border to the left and right edges of the page body

adds a 4-pixel double medium green border to the aside element

```
/* Page Body Styles */

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


/* Aside Styles */

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

Creating Rounded Corners

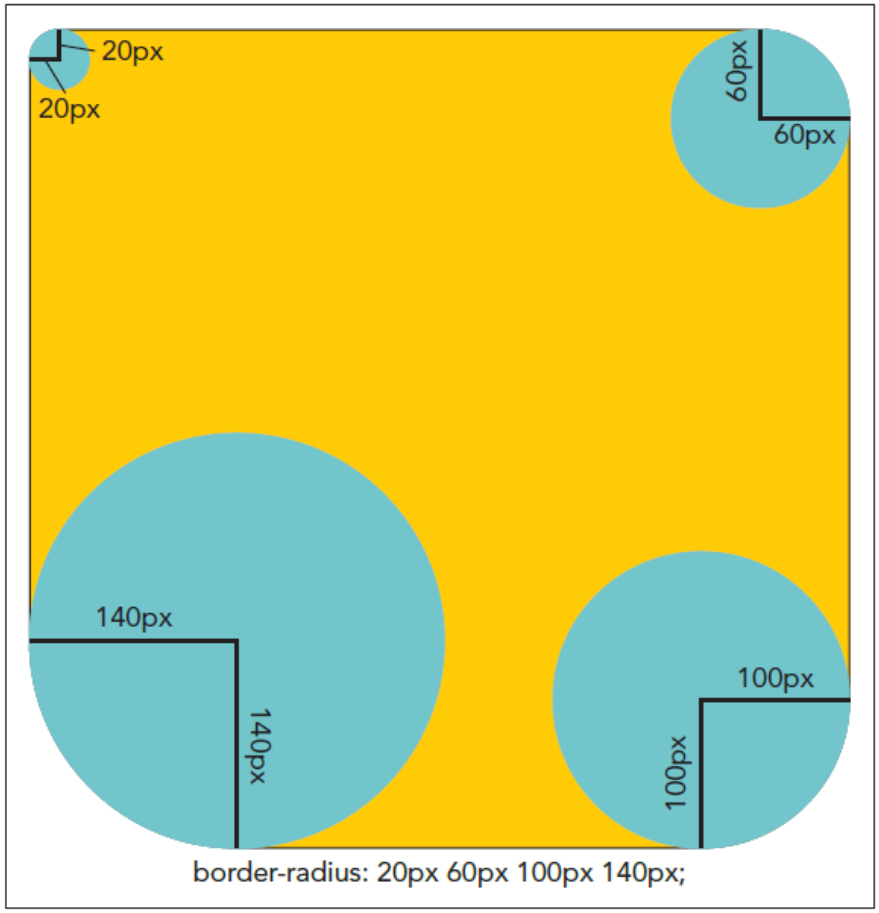
- Any of the four corners of a border can be rounded off by applying the following property:

```
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

Creating Rounded Corners (continued 1)

Figure 4-15 Setting rounded corners based on corner radii



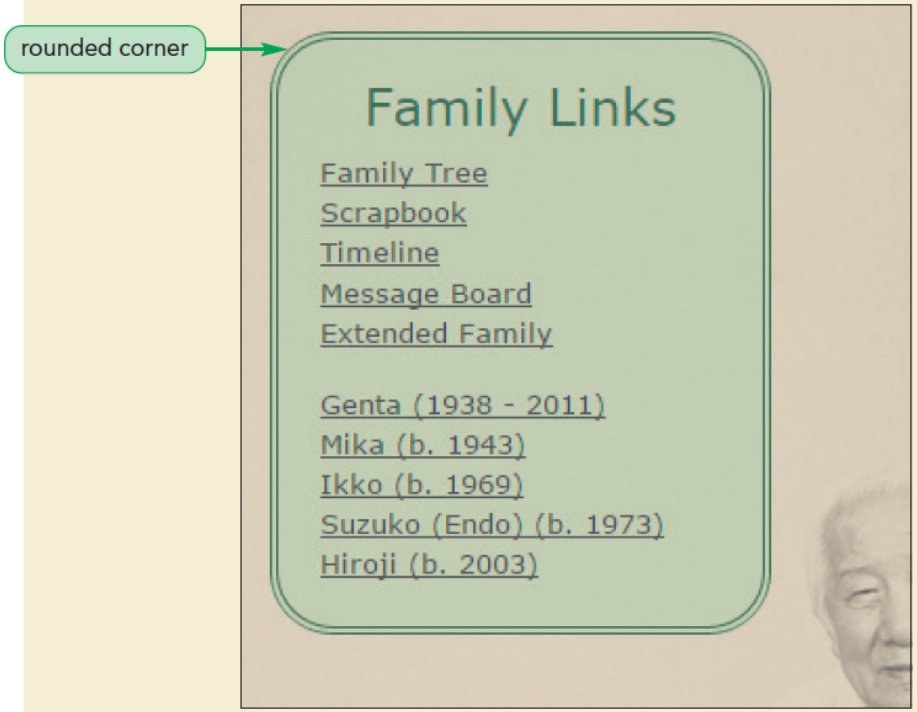
Creating Rounded Corners (continued 2)

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

sets the radius at each border corner to 30 pixels

```
aside {
   border: 4px double rgb(45, 93, 62);
   border-radius: 30px;
}
```

Figure 4-18 Aside element border with rounded corners



Applying a Border Image

- A border image is a border 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 content of the object appears in the interior piece and this piece is discarded

Applying a Border Image (continued 1)

- The four corners become the corners of the border
- The four sides are either stretched or tiled to fill in the border's top, right, bottom, and left sides

Applying a Border Image (continued 2)

- A border image is applied using the following property:

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

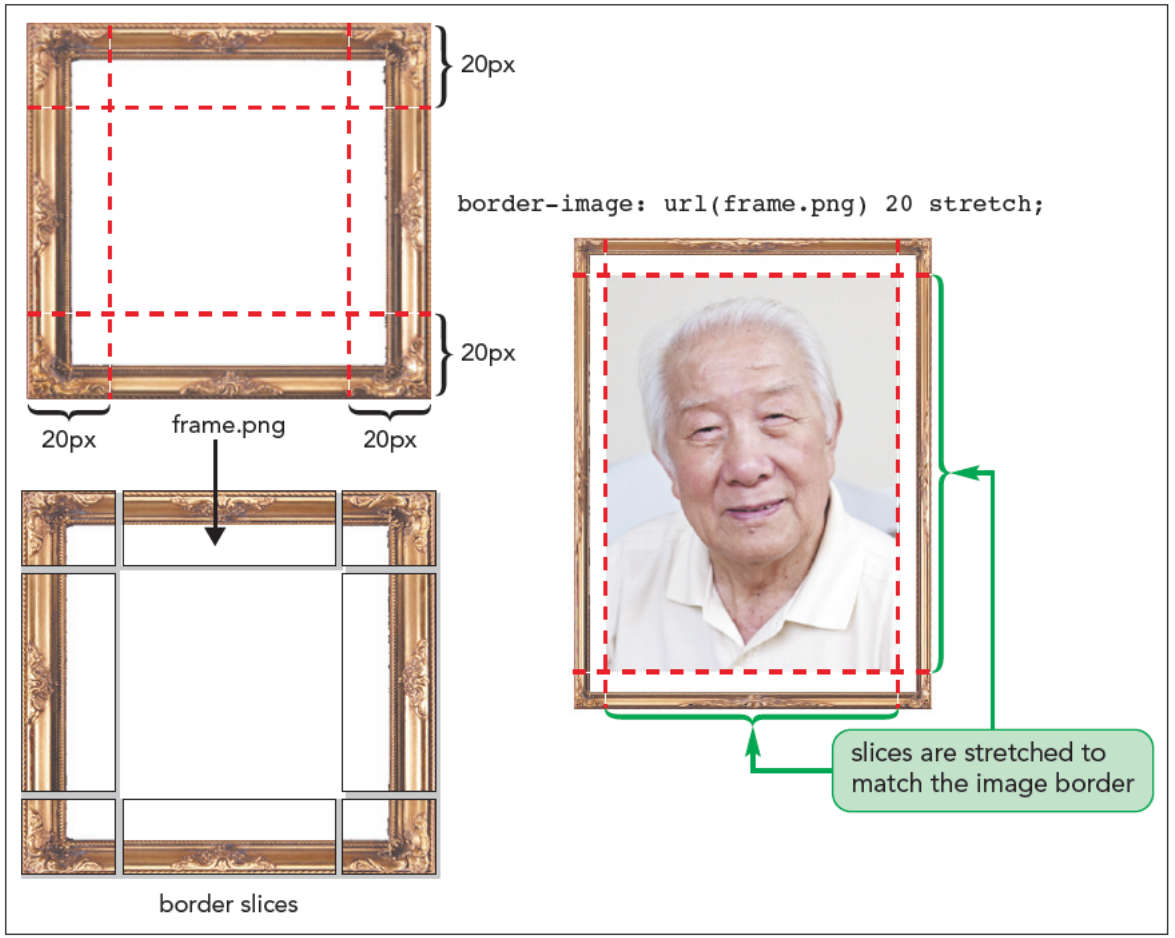
  - *url* indicates the source of the graphic image
  - *slice* indicates the width or height of the slice used to create the sides and corners
  - *repeat* indicates whether the side slices should be stretched or tiled to cover the four sides of the border

Applying a Border Image (continued 3)

- 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`: When the slices are tiled to fill each side, if they do not fill the sides with an integer number of tiles, the slices are rescaled until they do
  - `space`: When the slices are tiled to fill each side, if they do not fill the sides with an integer number of tiles, extra space is distributed around the tiles

Applying a Border Image (continued 4)

Figure 4-19 Slicing a graphic image to create a border



Creating a Text Shadow

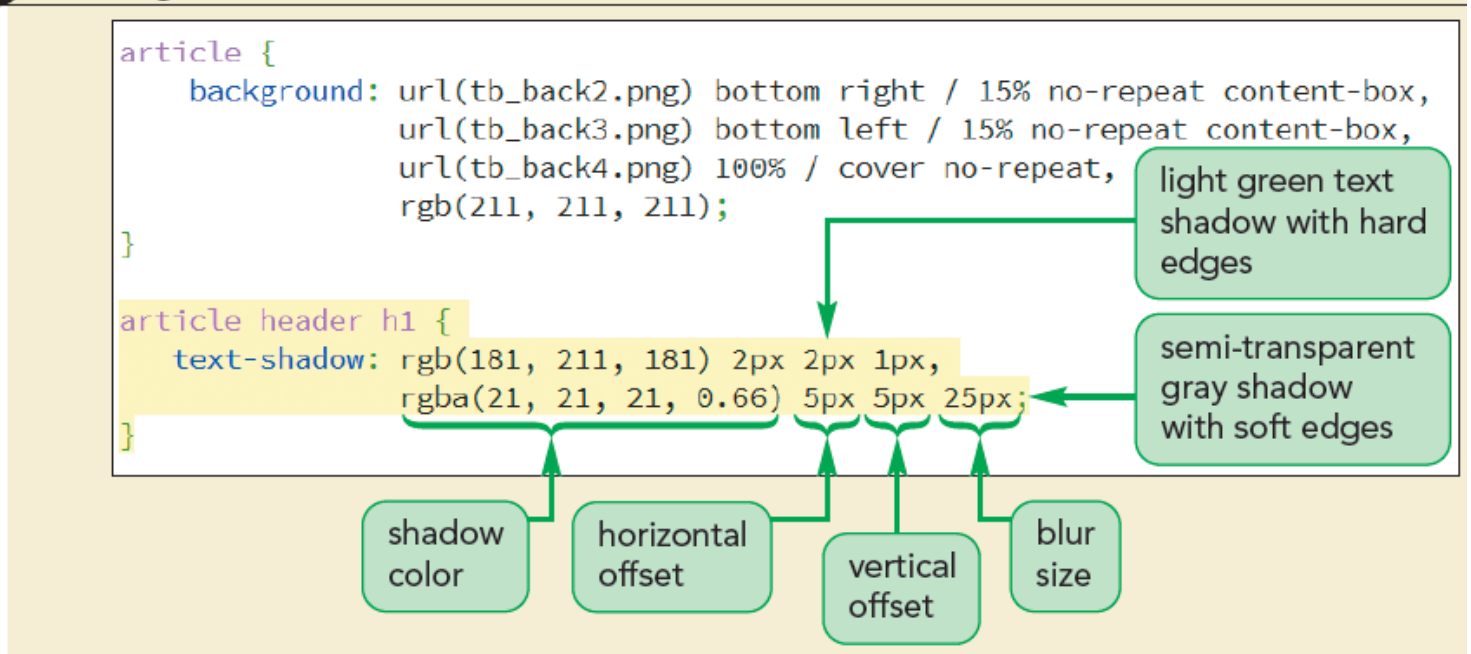
- A shadow can be added to a text on a page, to give the text a visual impact, by using the following property:

  `text-shadow: color offsetX offsetY blur;`

  - *color* is the shadow color
  - *offsetX* and *offsetY* are the distances of the shadow from the text in the horizontal and vertical directions
  - *blur* creates a blurred effect by spreading out a shadow

Creating a Text Shadow (continued)

Figure 4-23 Adding text shadows



Creating a Box Shadow

- Any block element can be shadowed 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
- Multiple shadows can be added by including them in a comma-separated list

Creating a Box Shadow (continued 1)

Figure 4-25 Adding box shadows

drop shadow on the page body's right edge

drop shadow on the page body's left edge

```
body {
   border-left: 1px solid rgb(51, 51, 51);
   border-right: 1px solid rgb(51, 51, 51);
   box-shadow: rgb(51, 51, 51) 15px 0px 25px,
               rgb(51, 51, 51) -15px 0px 25px;
}
```

gray shadow color

Figure 4-26 Page body with drop shadows

Creating a Box Shadow (continued 2)

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;
}
```

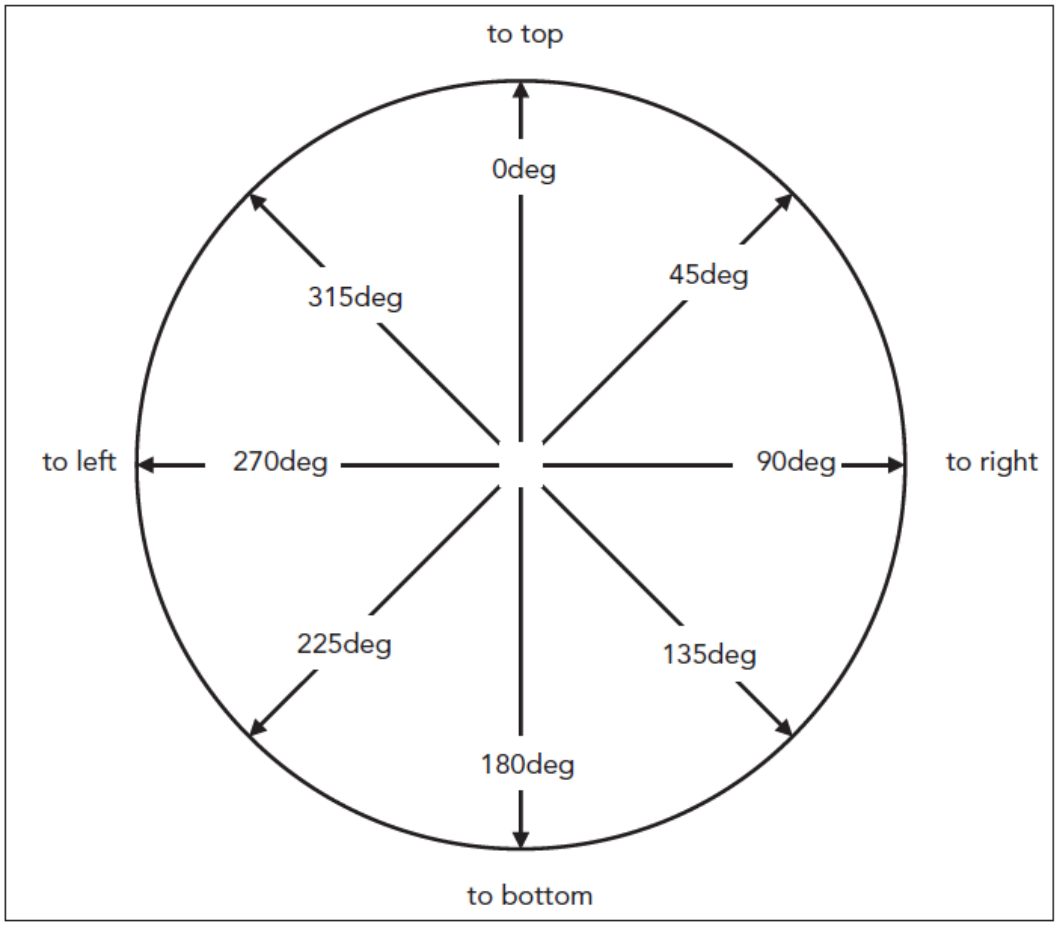
places a medium-gray shadow in the upper-left interior corner

Applying a Color Gradient

- **Color gradient:**
  - One color gradually blends into another color or fades away if transparent colors are used
  - It can be used to modify a background color
- **Linear gradient:**
  - Background colors transition from a starting color to an ending color along a straight line
  - The default direction is vertical, starting from top and moving to bottom

Applying a Color Gradient (continued)

Figure 4-33 Linear gradient directions



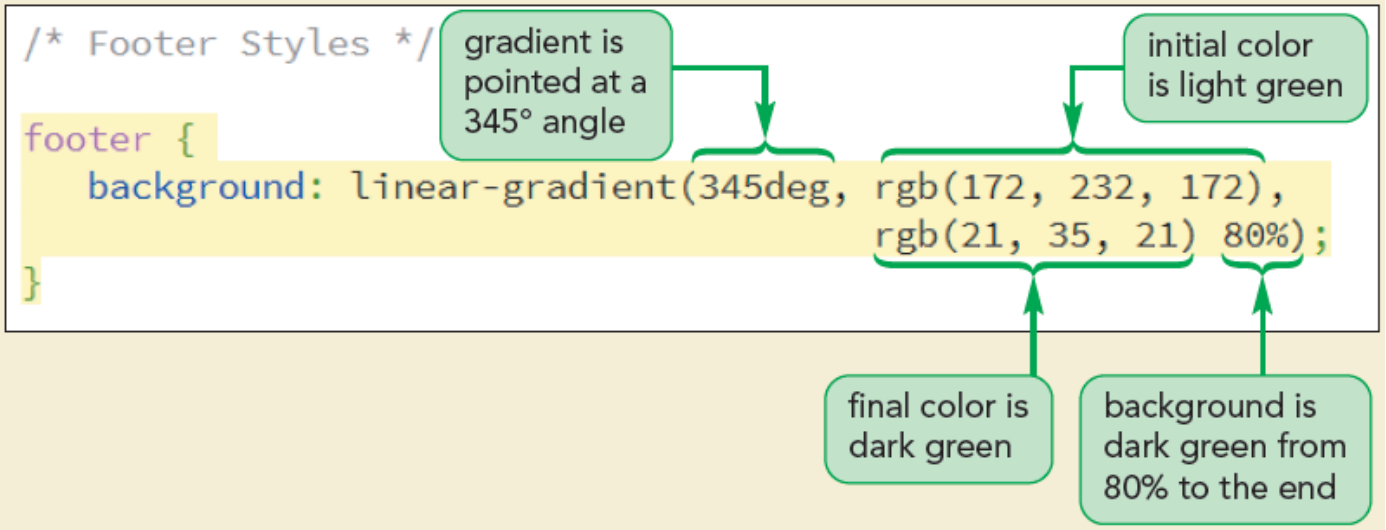
Gradients and Color Stops

- The colors specified in a gradient are evenly distributed
- 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)
```

Gradients and Color Stops (continued)

Figure 4-36 Applying a linear gradient



Creating a Radial Gradient

- **Radial gradient:**
  - It is a color gradient that starts from a central point
  - It proceeds outward in a series of concentric circles or ellipses

Creating a Radial Gradient (continued 1)

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

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

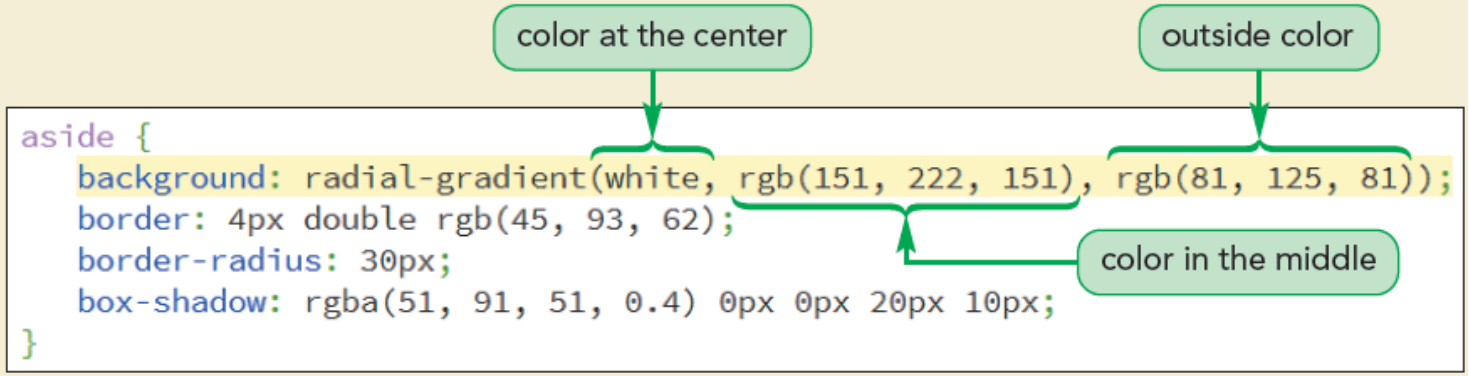
  - *`shape`* defines the shape of the gradient
  - *`position`* defines where the gradients radiate from
  - *`color-stop1, color-stop2,`* … specify the colors and their stopping positions

Creating a Radial Gradient (continued 2)

- The *size* value in the `radial-gradient` function:
  - defines the extent of the gradient as it radiates outward
  - can be expressed with a CSS unit of measure
  - can be expressed as a percentage of the background's width and height
  - can also be expressed with one of the following keywords: `farthest-corner` (the default), `farthest-side`, `closest-corner`, and `closest-side`

Creating a Radial Gradient (continued 3)

Figure 4-40 Applying a radial gradient



Repeating a Gradient

- A gradient design can be repeated to avoid the gradient function from being complicated due to the addition of more color stops

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

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

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

Creating Semi-Transparent Objects

- Semi-Transparent colors can be created using the `opacity` property

Figure 4-43 Creating a semi-transparent object

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

sets the opacity of the figure box to 55%

Figure 4-44 Changing the opacity of the figure box

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

Transforming Page Objects

- Rotation, rescaling, and translation in space can be used to transform the appearance of page objects

  `transform: effect(params);`

  - *effect* is the transformation function that will be applied to page objects
  - *params* specify the parameters required by the transformation function

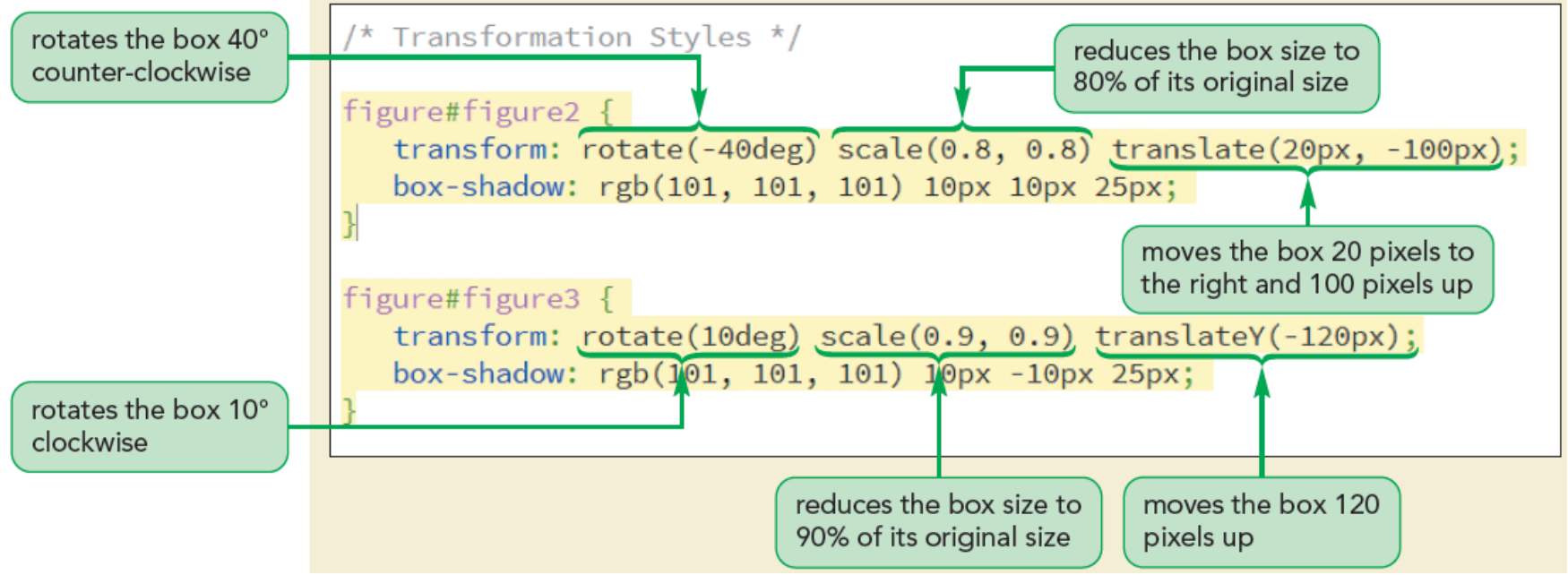
Transforming Page Objects (continued 1)

Figure 4-45 CSS3 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

Transforming Page Objects (continued 2)

Figure 4-48 Transforming the figure boxes



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
  - a *z*-axis that comes straight out of the page toward and away from the viewer

Transformations in Three Dimensions (continued)

Figure 4-51 CSS3 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

Understanding Perspective

- Perspective: A measure of how rapidly objects appear to recede from the viewer in a 3D space
  - It is used when only one object needs to be transformed in the 3D space
- Perspective can be thought in terms of a pair of railroad tracks that appear to converge at a point, known as the **vanishing point**

Understanding Perspective (continued 1)

- The perspective of a 3D space can be defined using

```
perspective: value;
```

  - *value* is a positive value that measures the strength of the perspective effect
  - Lower *value* results in more extreme distortion

Understanding Perspective (continued 2)

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;
}
```

Exploring CSS filters

- Filters adjust how a browser renders an image, a background, or a border
- Filters modify an object's color, brightness, contrast, or general appearance
- Filters were originally introduced as a WebKit browser extension

Exploring CSS filters (continued 1)

- Filters can be applied using the following property:

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

  - *effect* is a filter function
  - *params* specify the parameters of filter function

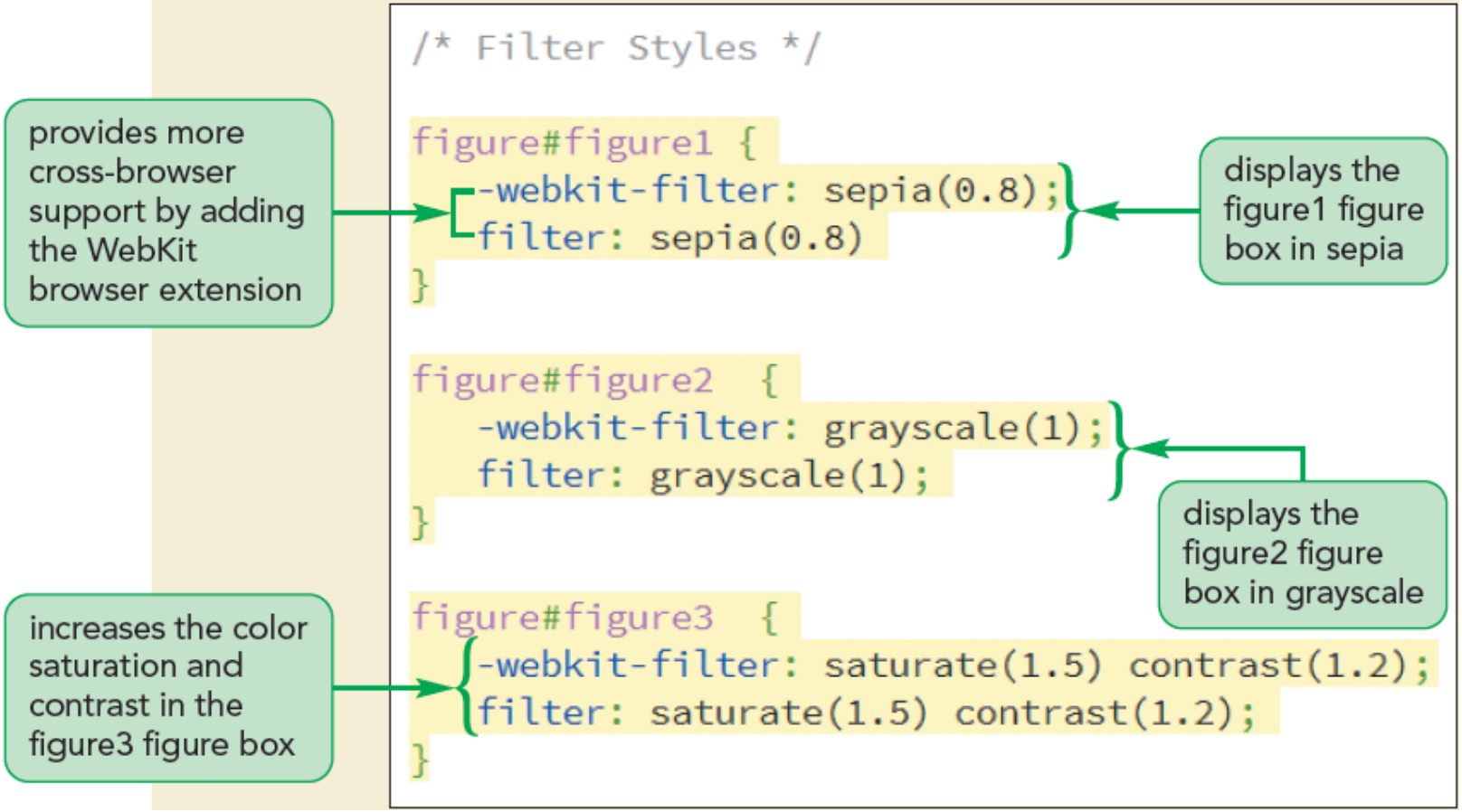
Exploring CSS filters (continued 2)

Figure 4-55 CSS3 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*

Exploring CSS filters (continued 3)

Figure 4-57 Applying the filter property



Working with Image Maps

- When an inline image is marked as a hyperlink, the entire image is linked to the same file
- HTML allows an image to be divided into different zones, or **hotspots**
- **Hotspots** can be linked to different URLs through information given in an **image map**

Working with Image Maps (continued)

- Image maps supported by HTML:
  - **Client-side image map** – Image map defined within a web page and handled entirely by the web browser
  - **Server-side image map** – Image map that relies on a program running on the web server to create and administer the map

Defining a Client-Side Image Map

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

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

  - *text* is the name of the image map
  - *hotspots* are the defined regions within an image that are linked to different URLs

Defining a Client-Side Image Map (continued 1)

- They can be placed anywhere within the body of a web page
- They are not actually displayed by browsers
- They 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 image

Defining a Client-Side Image Map (continued 2)

- Hotspot within the `map` element can be defined using the following element:

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

  - *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
  - *text* is alternate text displayed for non-graphical browsers

Defining a Client-Side Image Map (continued 3)

- 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 4-62 Inserting an image map

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

name of the image map

shape of the hotspot

coordinates of the rectangular hotspot

URL of the hotspot link

alternate text for the hotspot

Applying an Image Map

- An image map can be applied to an image using 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

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>
```