HTML5, CSS3, and JavaScript

6th Edition

Tutorial 2

Getting Started with CSS

Carey

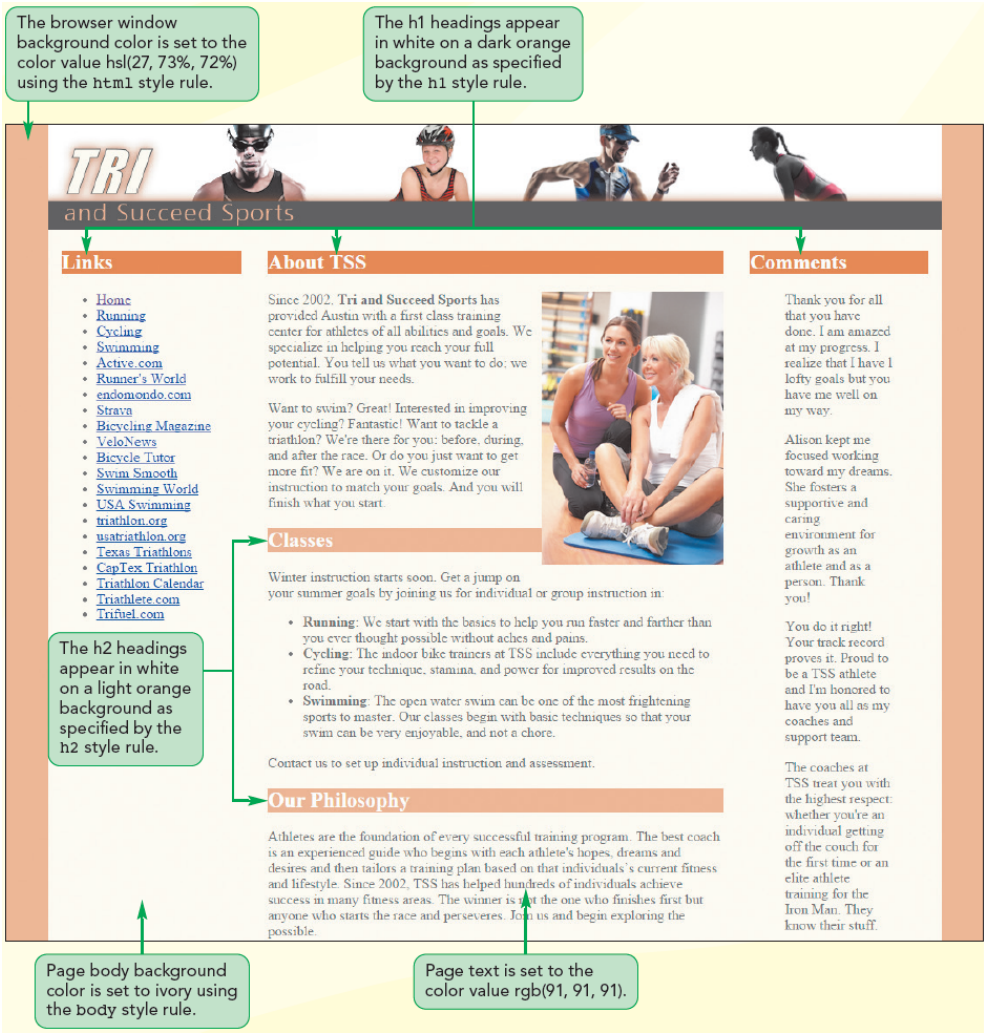
Objectives

- Explore the history of CSS
- Study different types of style sheets
- Explore style precedence and inheritance
- Apply color in CSS
- Use contextual selectors
- Work with attribute selectors

Objectives (continued)

- Apply text and font styles
- Use a web font
- Define list styles
- Work with margins and padding space
- Use pseudo-classes and pseudo-elements
- Insert page content with CSS

CSS Styles and Colors



Introducing CSS

- The appearance of the page is determined by one or more style sheets written in the Cascading Style Sheets (CSS) language
- The latest version of the CSS language is **CSS3**
- CSS3 is built upon several **modules**, where each module is focused on a separate design topic

Types of Style Sheets

- **Browser styles** or **user agent styles** – Styles built into the browser
- **User-defined styles** – Styles defined by a user based on the configuration setting of the user's browser
- **External styles** – Styles created by a website author, placed within a CSS file, and linked to the page

Types of Style Sheets (continued)

- **Embedded styles** – Styles added to the head of an HTML document
- **Inline styles** – Styles added as element attributes within an HTML document and applied to only that particular element

Exploring Style Rules

- The general syntax of a CSS style rule is

```
selector{
     property1: value1;
     property2: value2;
     ...
}
```

Exploring Style Rules (continued)

- **Browser extensions** are an extended library of style properties in the browser
- **Vendor prefix** – Indicates the browser vendor that created and supports the style property

Vendor prefixes for browser extensions

Vendor Prefix	Rendering Engine	Browsers
`-khtml-`	KHTML	Konqueror
`-moz-`	Mozilla	Firefox, Camino
`-ms-`	Trident	Internet Explorer
`-o-`	Presto	Opera, Nintendo Wii browser
`-webkit-`	WebKit	Android browser, Chrome, Safari

Embedded Style Sheets

- They are inserted directly into the HTML file as metadata by adding the following element to the document head

```
<style>
   style rules
</style>
```

where ***style rules*** are the different rules embedded in the HTML page

Inline Styles

- They are styles applied directly to specific elements using the following **style** attribute

```
<element style="property1: value1;
                property2: value2; …">
    content
</element>
```

where the *property: value* pairs define the styles applied directly to that element

Style Specificity and Precedence

- The more specific style rule has precedence over the more general style rule
- Specificity is an issue when two or more styles conflict
- If two rules have equal specificity and equal importance, then the one that is defined last has precedence

Style Inheritance

- **Style inheritance** – Process in which properties are passed from a parent element to its children
- For example, the following style rule sets the color of article text to blue and the rule is passed to any paragraph or other elements nested within that article

```
article {color: blue;}
p {text-align: center;}
```

Browser Developer Tools

- They allow designers to view HTML code and CSS styles
- They make it easier to locate the source of a style that has been applied to a specific page element
- They are different in each browser and are updated and improved constantly

Writing Style Comments

Adding the @charset rule and style comments

the charset rule defines the character encoding used in the style sheet

CSS comments provide information about the style sheet

author name and current date

```
@charset "utf-8";

/*
   New Perspectives on HTML5 and CSS3, 7th Edition
   Tutorial 2
   Tutorial Case

   TSS Typographic Style Sheet
   Author: Alison Palmer
   Date:   2017-03-01

   Filename: tss_styles.css

*/
```

Importing Style Sheets

- **@import** is a CSS at-rule used to import the content of a style sheet file

  `@import url(`*`url`*`);`

  where *`url`* is the URL of an external stylesheet file

- It is similar to adding `link` elements to an HTML file

Working with Color in CSS

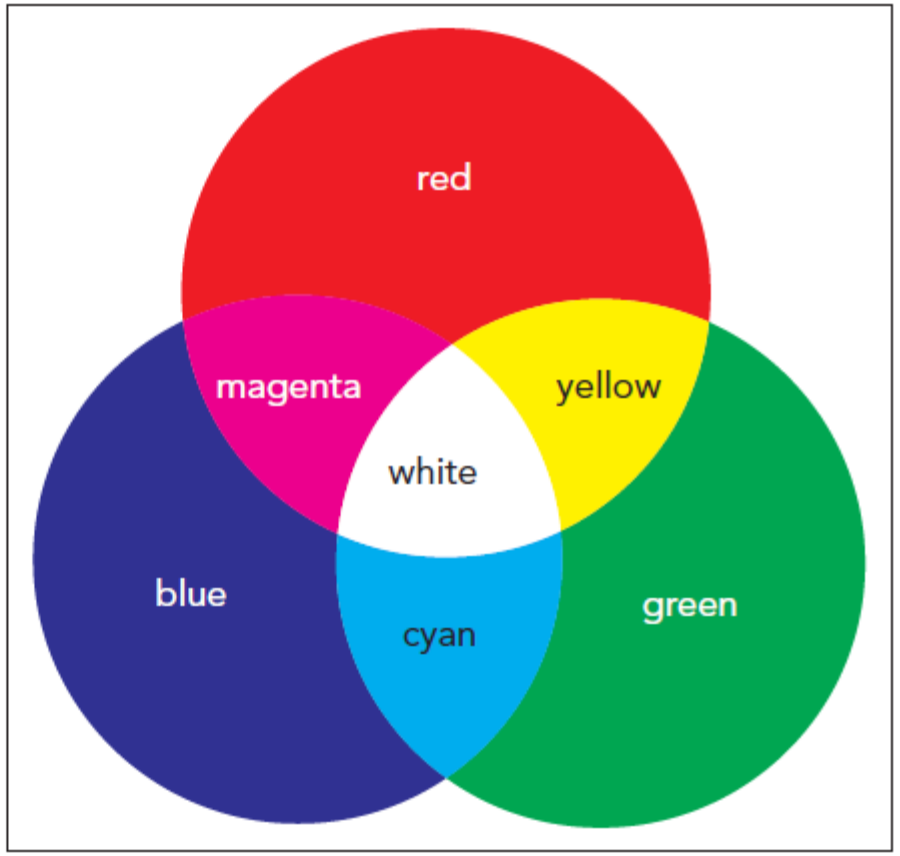
- **Color values** – Values in which the color is given by an exact numeric representation
- **RGB triplet** – The intensity of primary colors expressed as a set of numbers in CSS

```
rgb(red, green, blue)
```

- **Hexadecimal numbers** – A number expressed in the base 16 numbering system

RGB Color Values

Figure 2-8 Color addition in the RGB color model

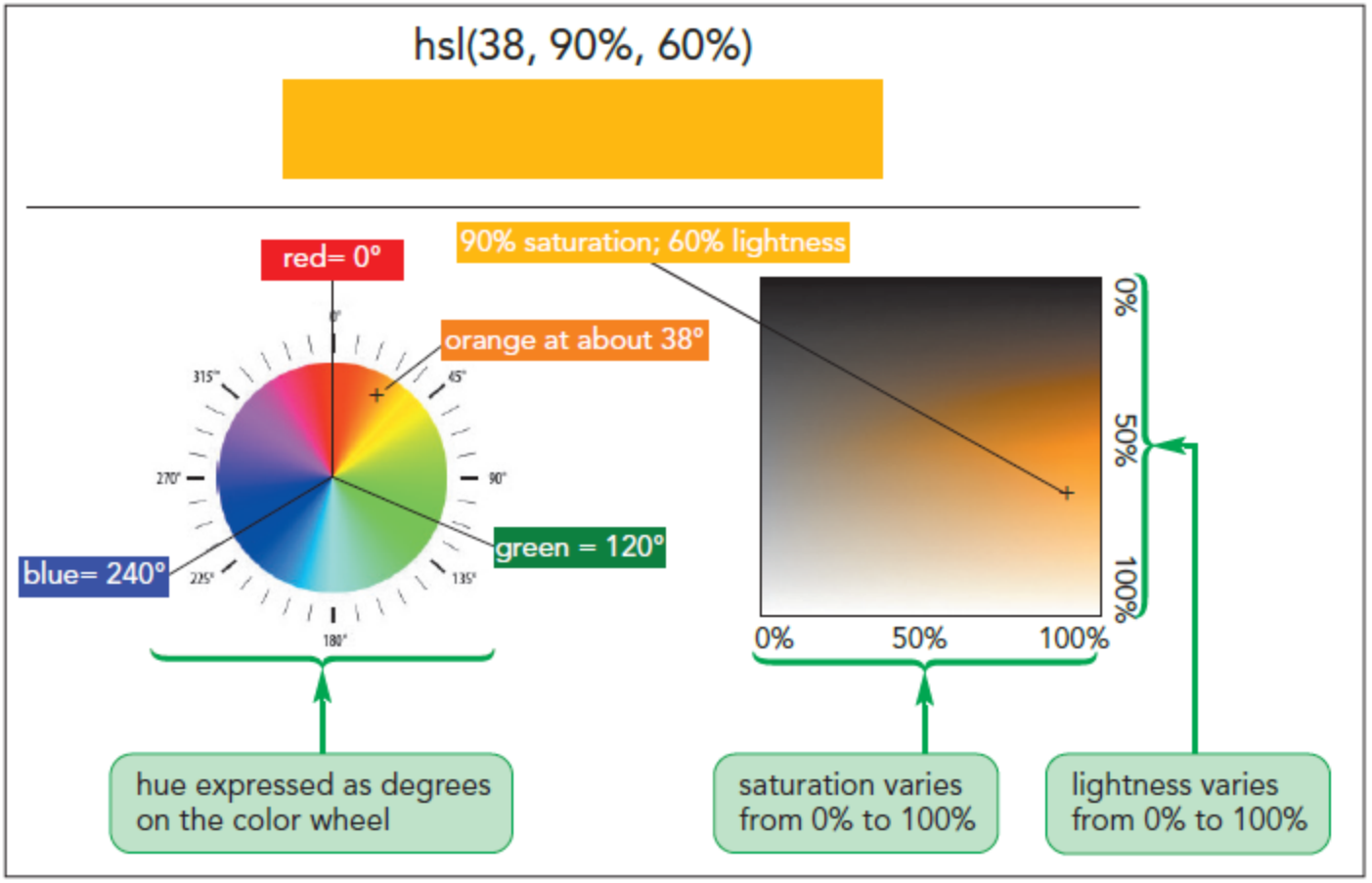


HSL Color Values

- **Hue** – Tint of a color, represented by a direction on a color wheel
- **Saturation** – Measures the intensity of a color and ranges from 0% (no color) up to 100% (full color)
- **Lightness** – Measures the brightness of a color and ranges from 0% (black) up to 100% (white)

HSL Color Values (continued)

Figure 2-9 Defining the color orange under the HSL color model



Defining Semi-Opaque Colors

- **Opacity** – Defines how solid a color appears
- A color's opacity is specified using the following properties:
  - `rgba(red, green, blue, opacity)`
  - `hsla(hue, saturation, lightness, opacity)`

  where `opacity` sets the opacity of the color ranging from 0 (completely transparent) up to 1.0 (completely opaque)

Setting Text and Background Color

- CSS defines the text and background color for each element on a webpage

```
color: color;
background-color: color;
```

where *color* is a color name or a color value

Employing Progressive Enhancement

- **Progressive enhancement** – A technique of placing the code conforming to elder standards before newer properties
- It provides support for older browsers and allows newer standards to be used by the browsers that support them

Contextual Selectors

- **Contextual selector** – Specifies the context under which a particular page element is matched
- Context is based on the hierarchical structure of a document, whichs involves the relationships between a **parent element** containing one or more **child elements** and within those child elements several levels of **descendant elements**

Contextual Selectors (continued 1)

Contextual selectors

Selector	Description
*	Matches any element
elem	Matches the element *elem* located anywhere in the document
elem1, elem2, …	Matches any of the elements *elem1*, *elem2*, etc.
parent descendant	Matches the *descendant* element that is nested within the *parent* element at some level
parent > child	Matches the *child* element that is a child of the *parent* element
elem1 + elem2	Matches *elem2* that is immediately preceded by the sibling element *elem1*
elem1 ~ elem2	Matches *elem2* that follows the sibling element *elem1*

Contextual Selectors (continued 2)

- To match any element, a **wildcard selector** with the * character is used
- **Sibling selectors** are used to select elements based on elements that are adjacent to them in the document hierarchy

Attribute Selectors

- Selectors also can be defined based on attributes and attribute values within elements
  - **`id`** – Identifies specific elements within the document
  - **`class`**– Identifies a group of elements that share a similar characteristic or property

Attribute Selectors (continued)

Figure 2-15 Attribute selectors

Selector	Selects	Example	Selects
elem#*id*	Element *elem* with the ID value *id*	h1#intro	The h1 heading with the id *intro*
#*id*	Any element with the ID value *id*	#intro	Any element with the id *intro*
elem.*class*	All *elem* elements with the class attribute value *class*	p.main	All paragraphs belonging to the *main* class
.*class*	All elements with the class value *class*	.main	All elements belonging to the *main* class
elem[*att*]	All *elem* elements containing the *att* attribute	a[href]	All hypertext elements containing the href attribute
elem[*att*="*text*"]	All *elem* elements whose *att* attribute equals *text*	a[href="top.html"]	All hypertext elements whose href attribute equals *top.html*
elem[*att*~="*text*"]	All *elem* elements whose *att* attribute contains the word *text*	a[rel~="glossary"]	All hypertext elements whose rel attribute contains the word *glossary*
elem[*att*|="*text*"]	All *elem* elements whose *att* attribute value is a hyphen-separated list of words beginning with *text*	p[id|="first"]	All paragraphs whose id attribute starts with the word *first* in a hyphen-separated list of words
elem[*att*^="*text*"]	All *elem* elements whose *att* attribute begins with *text* [***CSS3***]	a[rel^="prev"]	All hypertext elements whose rel attribute begins with *prev*
elem[*att*$="*text*"]	All *elem* elements whose *att* attribute ends with *text* [***CSS3***]	a[href$="org"]	All hypertext elements whose href attribute ends with *org*
elem[*att**="*text*"]	All *elem* elements whose *att* attribute contains the value *text* [***CSS3***]	a[href*="faq"]	All hypertext elements whose href attribute contains the text string faq

Working with Fonts

- **Typography** is the art of designing the appearance of characters and letters on a page
- Color and font are one of few properties in the CSS family of typographical styles

Choosing a Font

- Text characters are based on **fonts** that define the style and appearance of each character in the alphabet
- The general structure of defining font for any page element is

  `font-family: fonts;`

  where *fonts* is a comma-separated list,

  also known as a **font stack**

Choosing a Font (continued)

- **Specific font** – Identified by name and based on a font definition file stored in a user's computer or accessible on the web
- **Generic font** – Describes the general appearance of the characters in the text but does not specify any particular font definition file
  - Supports the font groups **serif, sans-serif, monospace, cursive, and fantasy**

Styling Web Page Text

Figure 2-19 Web safe font stacks

Exploring Web Fonts

- **Web font** – Definition font is supplied to the browser in an external file because web safe fonts limit the number of fonts choices

Figure 2-22 Web font formats

Format	Description	Browser
Embedded OpenType (EOT)	A compact form of OpenType fonts designed for use as embedded fonts in style sheets	IE
TrueType (TTF)	Font standard used on the Mac OS and Microsoft Windows operating systems	IE, Firefox, Chrome, Safari, Opera
OpenType (OTF)	Font format built on the TrueType format developed by Microsoft	IE, Firefox, Chrome, Safari, Opera
Scalable Vector Graphics (SVG)	Font format based on an XML vocabulary designed to describe resizable graphics and vector images	Chrome, Safari
Web Open Font Format (WOFF)	The W3C recommendation font format based on OpenType and TrueType with compression and additional metadata	IE, Firefox, Chrome, Safari, Opera

The @font-face Rule

- To access and load a web font, add the `@font-face` rule to the style sheet
- Once a web font is defined using the `@font-face` rule, it is included in the font stack

The @font-face Rule (continued 1)

- The general syntax to include `@font-face` is

```
@font-face {
   font-family: name;
   src: url (‘url1’) format
(‘text1’),
      url (‘url2’) format (‘text2’),
   …;
   descriptor1: value1;
   descriptor2: value2;
   …
      }
```

The @font-face Rule (continued 2)

where `name` is the name of the font, `url` is the location of the font definition file, `text` is an optional text description of the font format, and the `descriptor1: value1;` pairs are optional style properties of the font

Setting the Font Size

- To set a font size, use the style property

  ```
  font-size: size;
  ```

  where *size* is a CSS unit of length in either relative or absolute units.

- **Absolute units** – Fixed in size regardless of the output device and are used only with printed media
- **Relative units** – Expressed relative to the size of other objects within the web page or to the display properties of the device itself

Scaling Fonts with ems and rems

- Text is made scalable with all font sizes expressed relative to default font sizes
- The three relative measurements used to provide scalability are
  - percentage
  - em unit
  - rem or root em unit

Using Viewport Units

- **Viewport unit** – A relative unit used to express length as a percentage if the width and height if the browser window
- CSS3 introduced four viewport units
  - 1 vw = 1% of the browser window width
  - 1 vh = 1% of the browser window height
  - 1 vmin = 1 vw or 1 vh (whichever is smaller)
  - 1 vmax = 1 vw or 1 vh (whichever is larger)

Sizing Keywords

- Font sizes are expressed using the following keywords
  - `xx-small`
  - `x-small`
  - `small`
  - `medium`
  - `large`
  - `x-large`
  - `xx-large`
  - `larger`
  - `smaller`

Controlling Spacing and Indentation

- **Kerning** measures the amount of space between characters, while **tracking** measures the amount of space between words
- The properties to control an element's kerning and tracking are:

```
letter-spacing: value;
word-spacing: value;
```

Controlling Spacing and Indentation (continued 1)

- **Leading** – Measures the amount of space between lines of text and is set using the following line-height property:

```
line-height: size;
```

- Text spacing can be controlled by setting the indentation for the first line of text block by using the **text-indent** property

```
text-indent: size;
```

Controlling Spacing and Indentation (continued 2)

Figure 2-27 Controlling letter spacing and line height

```
h1, h2 {
   font-family: Quicksand, Verdana, Geneva, sans-serif;
   letter-spacing: 0.1em;
}

/* Navigation Styles */

nav {
   font-size: 0.8em;
}

nav > ul {
   line-height: 2em;
}
```

sets the space between letters to 0.1em

double spaces the list of hypertext links

Working with Font Styles

- To specify the font style, use

  `font-style: type;`

  where *type* is normal, italic, or oblique

- To change the weight of the font, use

  `font-weight: weight;`

  where *weight* is the level of bold formatting applied to the text

Working with Font Styles (continued 1)

- To specify a text decoration, use

  `text-decoration: type;`

  where *type* is none, underline, overline, or line-through

- To transform text, use

  `text-transform: type;`

  where *type* is capitalize, uppercase, lowercase, or none

Working with Font Styles (continued 2)

- To display a font variant of text, use

  `font-variant: type;`

  where *type* is normal or small-caps

Aligning Text Horizontally and Vertically

- To horizontally align the text , use

  `text-align: *alignment*;`

  where *`alignment`* is left, right, center, or justify

- To vertically align the text within each line, use

  `vertical-align: *alignment*;`

  where *`alignment`* is baseline, bottom, middle, sub, super, text-bottom, text-top, or top

Aligning Text Horizontally and Vertically (continued)

Figure 2-28 Values of the vertical-align property

Value	Description
`baseline`	Aligns the baseline of the element with the baseline of the parent element
`bottom`	Aligns the bottom of the element with the bottom of the lowest element in the line
`middle`	Aligns the middle of the element with the middle of the surrounding content in the line
`sub`	Subscripts the element
`super`	Superscripts the element
`text-bottom`	Aligns the bottom of the element with the bottom of the text in the line
`text-top`	Aligns the top of the element with the top of the text in the line
`top`	Aligns the top of the element with the top of the tallest object in the line

Combining All Text Formatting in a Single Style

- The text and font styles can be combined using the following shorthand font property:

```
font: style variant weight
size/height family;
```

where *style* is the font's style, *variant* is the font variant, *weight* is the font weight, *size* is the font size, *height* is the height of each line, and *family* is the font stack

Combining All Text Formatting in a Single Style (continued)

Figure 2-29 Style rule for the body footer

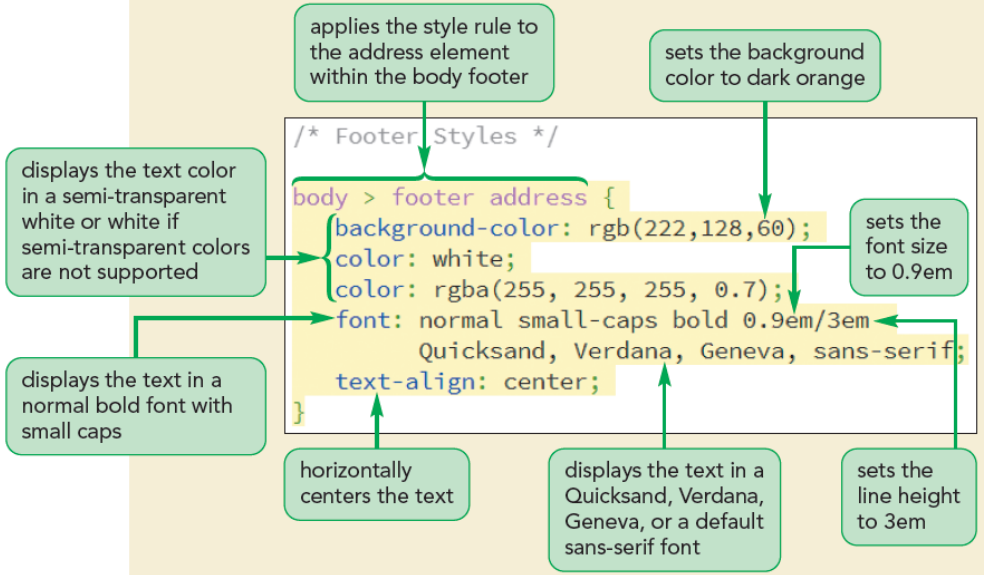
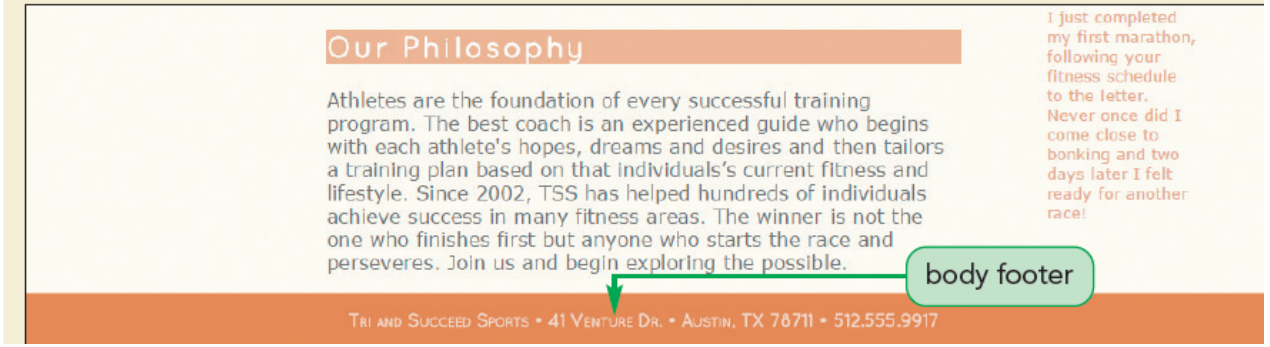


Figure 2-30 Formatted body footer



Formatting Lists

- **List marker** – It is the default browser style symbol displayed before each list item for unordered and ordered lists
- To change the type of list marker or to prevent any display of a list marker, use

  `list-style-type: type;`

  where *type* is the various types of markers

Formatting Lists (continued)

Figure 2-31 Values of the list-style-type property

list-style-type	Marker(s)
`disc`	●
`circle`	○
`square`	■
`decimal`	1, 2, 3, 4, …
`decimal-leading-zero`	01, 02, 03, 04, …
`lower-roman`	i, ii, iii, iv, …
`upper-roman`	I, II, III, IV, …
`lower-alpha`	a, b, c, d, …
`upper-alpha`	A, B, C, D, …
`lower-greek`	α, β, γ, δ, …
`upper-greek`	A, B, Γ, Δ, …
`none`	no marker displayed

Using Images for List Markers

- A customized graphic image for the list marker can be supplied by the user

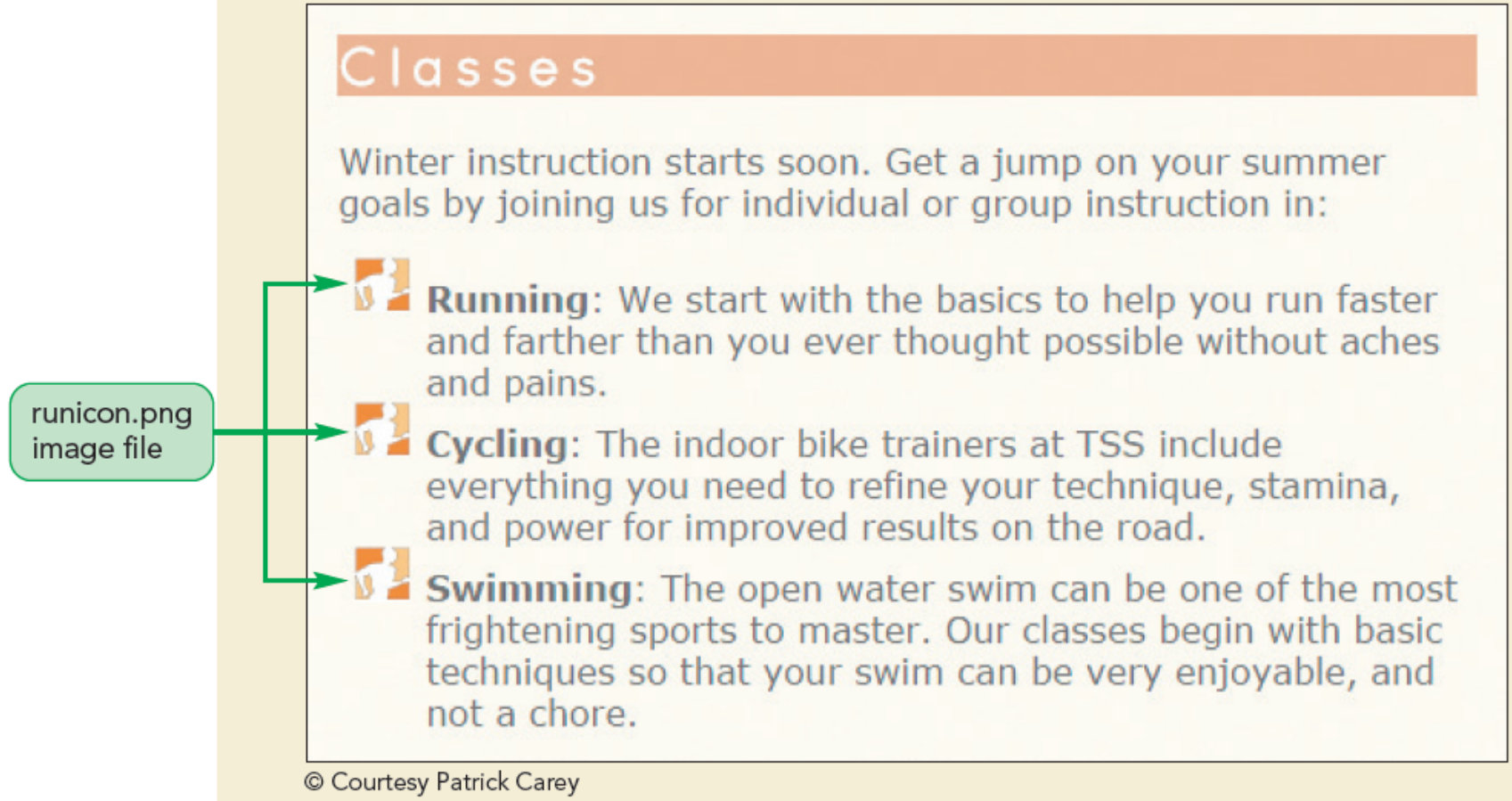
  `list-style-image: url(`*`url`*`);`

  where *`url`* is the URL of a graphic file containing the marker image

Figure 2-35 Displaying an image in place of a list marker

Using Images for List Markers (continued)

Figure 2-36 Unordered list with the runicon.png image marker



Setting the List Marker Position

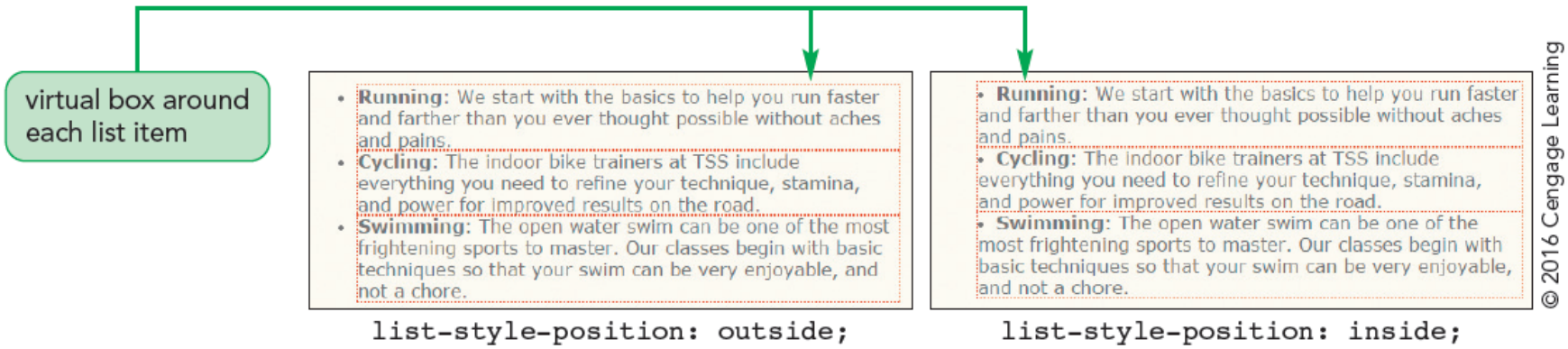
- CSS treats each list item as a block-level element, placed within a virtual box in which the list marker is placed outside of the list text
- To change the default behaviour, use

  `list-style-position: position;`

  where *position* is either outside or inside

Setting the List Marker Position (continued)

Figure 2-37 Values of the list-style-position property



Working with Margins and Padding

- Block-level elements follow the structure of the **box model**
- Contents in a **box model** are enclosed within the following series of concentric boxes:
  - The content of the element itself
  - The **padding space**, which extends from the element's content to a border

Working with Margins and Padding (continued)

- The **border** surrounding the padding space
- The **margin space** comprised of the space beyond the border up to the next page element

Figure 2-38 The CSS box model

Setting the Padding Space

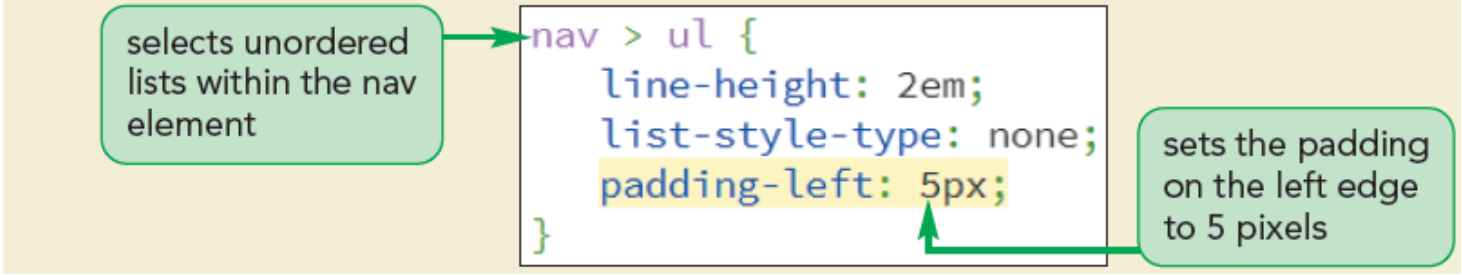
- To set the width of the padding space, use the following **padding** property

  `padding: size;`

  where *`size`* is expressed in one of the CSS units of length or the keyword `auto` to let the browser automatically choose the padding

Setting the Padding Space (continued)

Figure 2-39 Setting the size of the left padding space

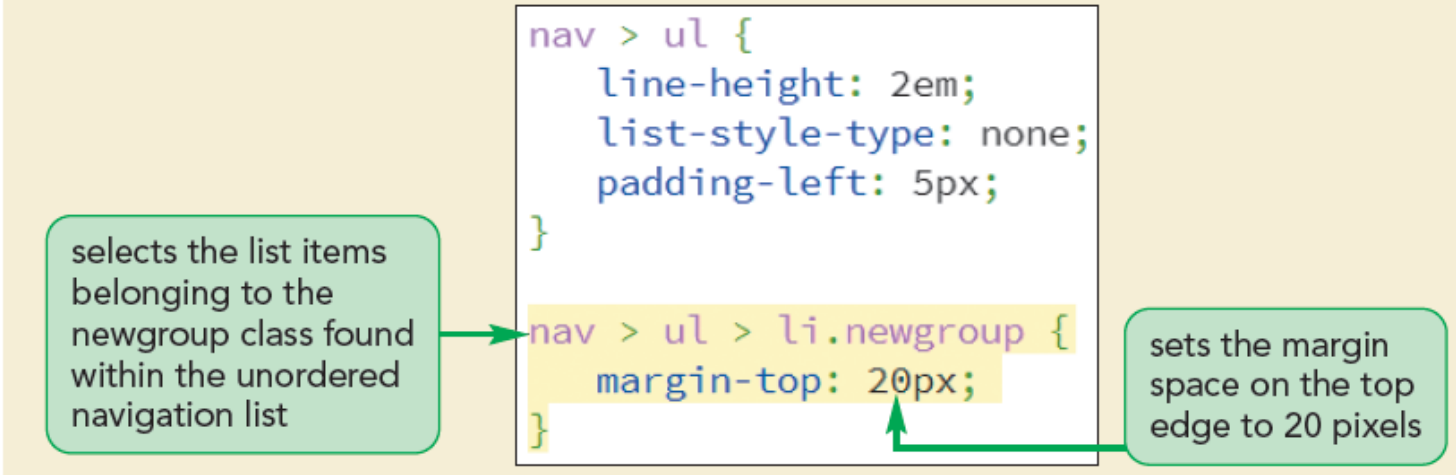


Setting the Margin and Border Spaces

- To set the size of the margin around block-level elements, use
  - `margin: size;`
  - `margin: top right bottom left;`
- To set the size of the border space, use
  - `border-width: size;`
  - `border-width: top right bottom left;`

Setting the Margin and Border Spaces (continued)

Figure 2-40 Setting the size of the top margin



Using Pseudo-Classes and Pseudo-Elements

- **Pseudo-class** – classifies an element based on its current status, position, or use in the document

  `element: pseudo-class`

  where `element` is an element from the document and `pseudo-class` is the name of a css pseudo-class

- **Structural pseudo-class** – classifies an element based on its location within the structure of the HTML document

Using Pseudo-Classes and Pseudo-Elements (continued 1)

Figure 2-43 Structural pseudo-classes

Pseudo-Class	Matches
`:root`	The top element in the document hierarchy (the `html` element)
`:empty`	An element with no content
`:only-child`	An element with no siblings
`:first-child`	The first child of the parent element
`:last-child`	The last child of the parent element
`:first-of-type`	The first descendant of the parent that matches the specified type
`:last-of-type`	The last descendant of the parent that matches the specified type
`:nth-of-type(`*n*`)`	The n^{th} element of the parent of the specified type
`:nth-last-of-type(`*n*`)`	The n^{th} from the last element of the parent of the specified type
`:only-of-type`	An element that has no siblings of the same type
`:lang(`*code*`)`	The element that has the specified language indicated by *code*
`:not(`*selector*`)`	An element not matching the specified *selector*

Using Pseudo-Classes and Pseudo-Elements (continued 2)

Figure 2-44 Applying pseudo-classes to list items

Pseudo-classes for Hypertext

- **Dynamic pseudo-class** – A type of pseudo-class in which the class can change state based on the actions of the user

Figure 2-46 Dynamic pseudo-classes

Pseudo-Class	Description
`:link`	The link has not yet been visited by the user.
`:visited`	The link has been visited by the user.
`:active`	The element is in the process of being activated or clicked by the user.
`:hover`	The mouse pointer is hovering over the element.
`:focus`	The element is receiving the focus of the keyboard or mouse pointer.

Pseudo-Elements

- **Pseudo-element** – An object that exists only in the rendered page
- Pseudo-elements can be selected using the following CSS selector:

  element::*pseudo-element*

  where *element* is an element from the HTML file and *pseudo-element* is the name of a CSS pseudo-element

Pseudo-Elements (continued)

Figure 2-49 Pseudo-elements

Pseudo-Element	Description
`::first-letter`	The first letter of the element text
`::first-line`	The first line of the element text
`::before`	Content inserted directly before the element
`::after`	Content inserted directly after the element

Generating Content with CSS

- New content can be added either before or after an element using the following **before** and **after** pseudo-elements:

```
element::before {content: text;}
element::after {content: text;}
```

where *text* is the content to be inserted into the rendered web page

Generating Content with CSS (continued)

Figure 2-50 Values of the content property

Value	Description
`none`	Sets the content to an empty text string
`counter`	Displays a counter value
`attr(attribute)`	Displays the value of the selector's *attribute*
text	Displays the specified *text*
`open-quote`	Displays an opening quotation mark
`close-quote`	Displays a closing quotation mark
`no-open-quote`	Removes an opening quotation mark, if previously specified
`no-close-quote`	Removes a closing quotation mark, if previously specified
`url(url)`	Displays the content of the media (image, video, etc.) from the file located at *url*

Displaying Attribute Values

- The content property can be used to insert an attribute value into the rendered web page using the **attr( )** function

```
content: attr(attribute);
```

where *attribute* is an attribute of the selected element

Inserting Quotation Marks

- The **blockquote** and **q** elements are used for quoted material
- Decorative opening and closing quotation marks can be inserted using the **content** property as follows:

```
content: open-quote;
content: close-quote;
```

Inserting Quotation Marks (continued)

Figure 2-51 Adding quotation marks to block quotes