CS 50 Web Design

APCSP Module 2: Internet

Unit 1: HTML (Chapter 5-7)

LECTURE 2: MARKUP LANGUAGE, LINK AND IMAGE

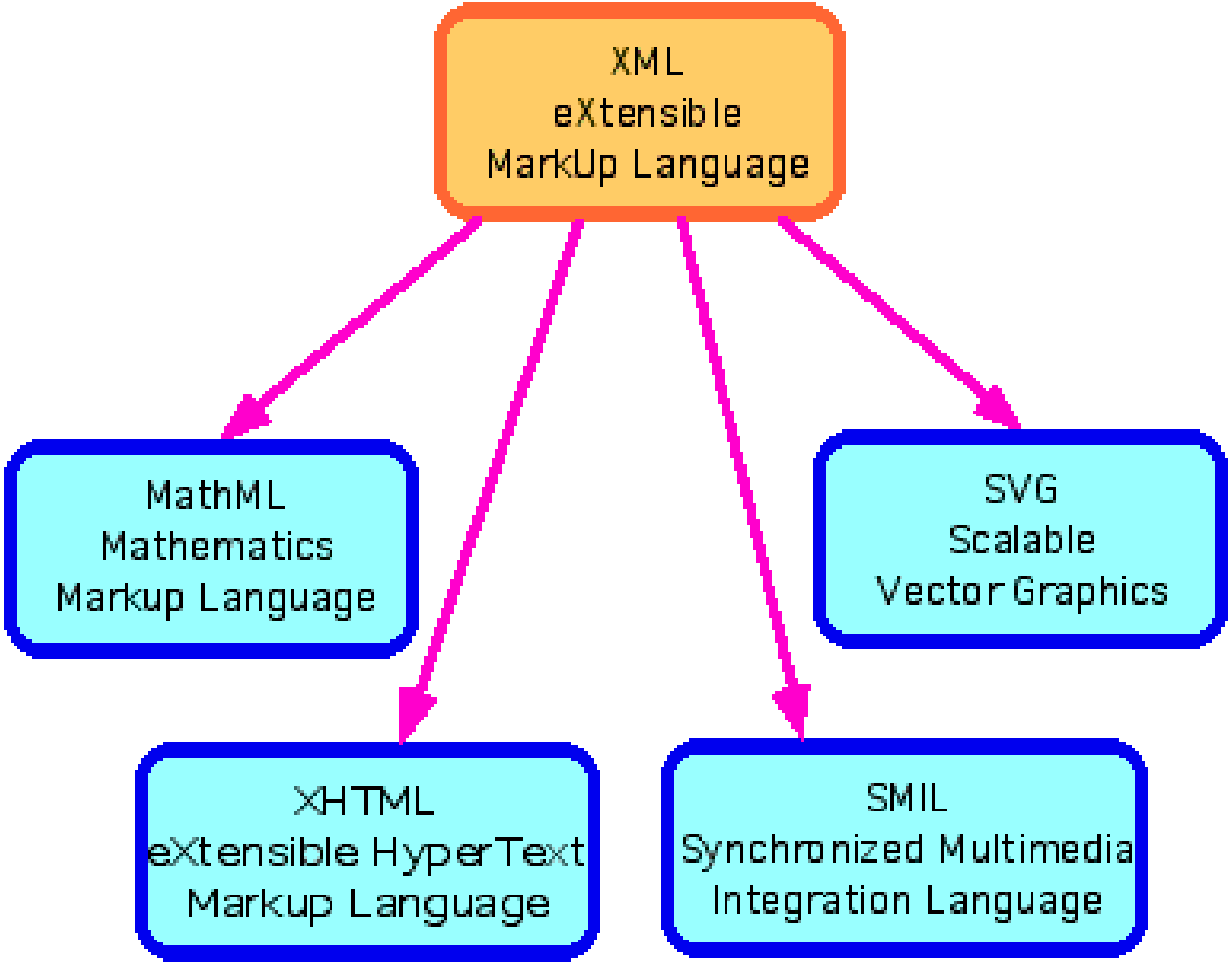
DR. ERIC CHOU IEEE SENIOR MEMBER

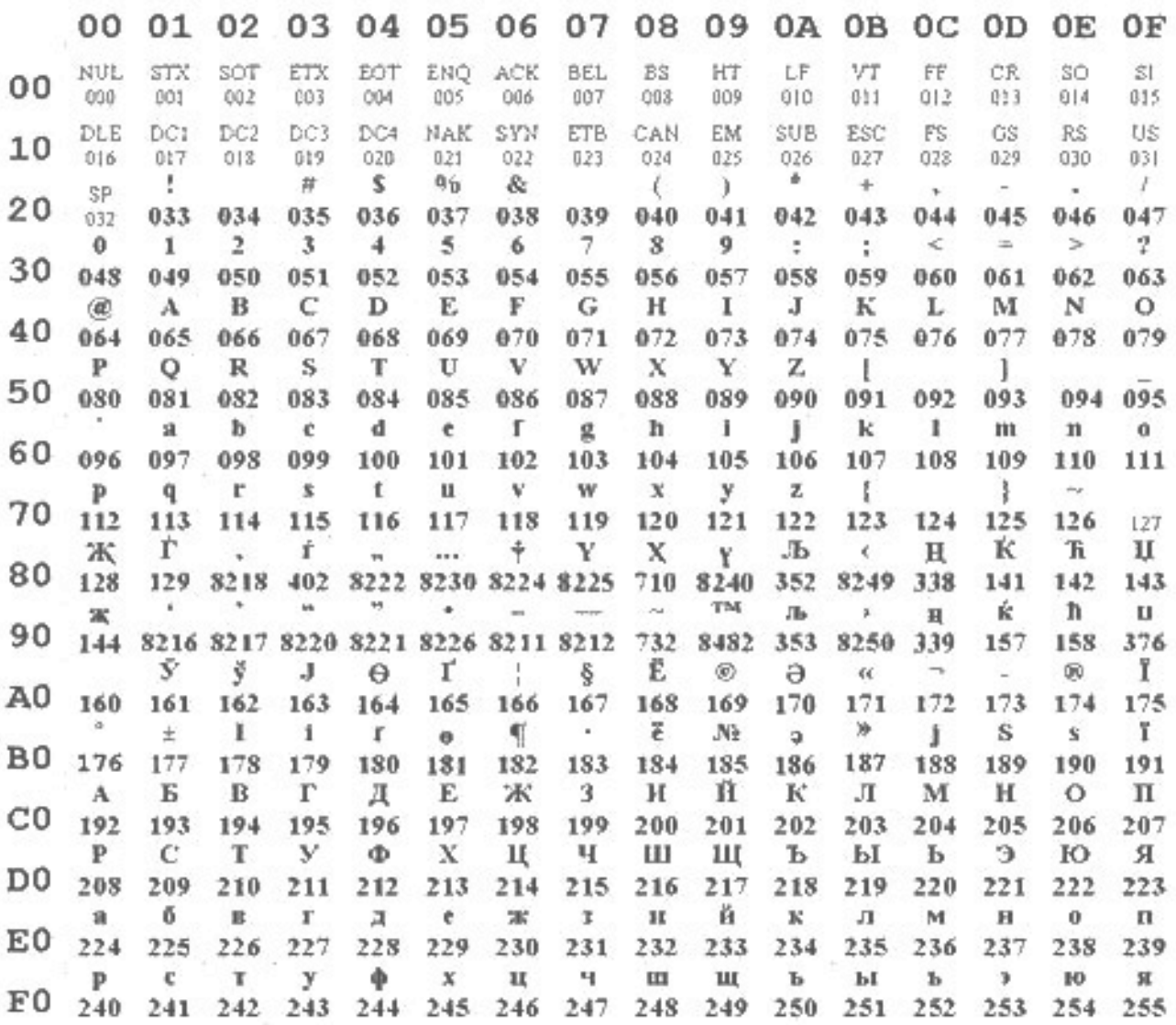
Basic Document Structure

SECTION 1

Markup language was used before Web Age

- Latex was a markup language to used to markup text file into a quality document which processes header, footer, font, bold, italics, subscript, superscript. (In fact, my own Ph.D. Dissertation was done in Latex) For more information regarding Latex, try www.latexeditor.org
- Hypertext Markup Language (HTML) is a markup language for hypertext which means text, image, video, audio, animation and many other multimedia component
- Micro-soft Word is also a markup authoring tool. Since 2013, Microsoft had started to use a new file format (docx), which means document file with XHTML enhancement.


XML
eXtensible
MarkUp Language
MathML
Mathematics
Markup Language
SVG
Scalable
Vector Graphics
XHTML
eXtensible HyperText
Markup Language
SMIL
Synchronized Multimedia
Integration Language



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 000	STX 001	SOT 002	ETX 003	EOT 004	ENQ 005	ACK 006	BEL 007	BS 008	HT 009	LF 010	VT 011	FF 012	CR 013	SO 014	SI 015
10	DLE 016	DC1 017	DC2 018	DC3 019	DC4 020	NAK 021	SYN 022	ETB 023	CAN 024	EM 025	SUB 026	ESC 027	FS 028	GS 029	RS 030	US 031
20	SP 032	! 033	034	# 035	$ 036	% 037	& 038	039	(040	) 041	* 042	+ 043	, 044	- 045	. 046	/ 047
30	0 048	1 049	2 050	3 051	4 052	5 053	6 054	7 055	8 056	9 057	: 058	; 059	< 060	= 061	> 062	? 063
40	@ 064	A 065	B 066	C 067	D 068	E 069	F 070	G 071	H 072	I 073	J 074	K 075	L 076	M 077	N 078	O 079
50	P 080	Q 081	R 082	S 083	T 084	U 085	V 086	W 087	X 088	Y 089	Z 090	[091	092	] 093	094	_ 095
60	` 096	a 097	b 098	c 099	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
70	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	124	} 125	~ 126	127
80	Ж 128	Ґ 129	‚ 8218	ƒ 402	„ 8222	… 8230	† 8224	‡ 8225	Ү 710	ү 8240	Љ 352	‹ 8249	Њ 338	Ќ 141	Ћ 142	Џ 143
90	ж 144	‘ 8216	’ 8217	“ 8220	” 8221	• 8226	– 8211	— 8212	~ 732	™ 8482	љ 353	› 8250	њ 339	ќ 157	ћ 158	џ 376
A0	160	Ў 161	ў 162	Ј 163	Ө 164	Ґ 165	¦ 166	§ 167	Ё 168	© 169	Э 170	« 171	¬ 172	- 173	® 174	Ї 175
B0	° 176	± 177	І 178	і 179	ґ 180	ө 181	¶ 182	· 183	ё 184	№ 185	э 186	» 187	ј 188	Ѕ 189	ѕ 190	ї 191
C0	А 192	Б 193	В 194	Г 195	Д 196	Е 197	Ж 198	З 199	И 200	Й 201	К 202	Л 203	М 204	Н 205	О 206	П 207
D0	Р 208	С 209	Т 210	У 211	Ф 212	Х 213	Ц 214	Ч 215	Ш 216	Щ 217	Ъ 218	Ы 219	Ь 220	Э 221	Ю 222	Я 223
E0	а 224	б 225	в 226	г 227	д 228	е 229	ж 230	з 231	и 232	й 233	к 234	л 235	м 236	н 237	о 238	п 239
F0	р 240	с 241	т 242	у 243	ф 244	х 245	ц 246	ч 247	ш 248	щ 249	ъ 250	ы 251	ь 252	э 253	ю 254	я 255

Character Set

<meta charset="utf-8"> tag

Used to pick utf-8 as the character set.

There are ASCII, extended ASCII, utf-8, utf-16, utf-32 character sets

Tags

SECTION 2

Text Content Tags (I)

Heading: <h1></h1>

/* heading of different level: h1, h2, h3,h4, …*/

Paragraph:

Unordered list: List Item 1

List Item 2

Ordered list: List Item 1

List Item 2

Description list: <dl><dt>Term</dt>

<dd>Definition</dd>

</dl>

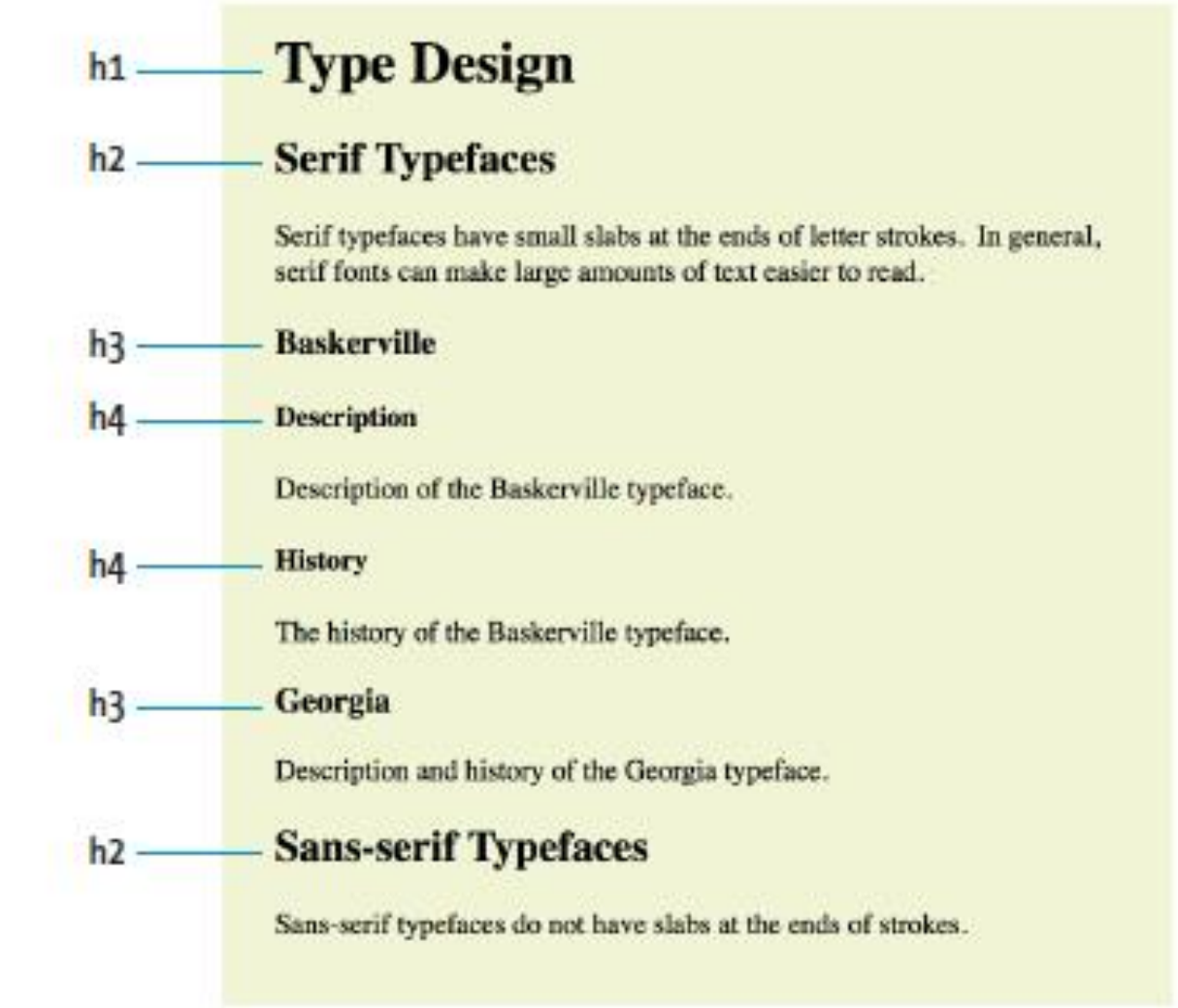


FIGURE 5-1. The default rendering of four heading levels.

Header Levels

```
<h3>Times</h3>
<p>Description and history of the Times typeface.</p>
<hr>
<h3>Georgia</h3>
<p>Description and history of the Georgia typeface.</p>
```

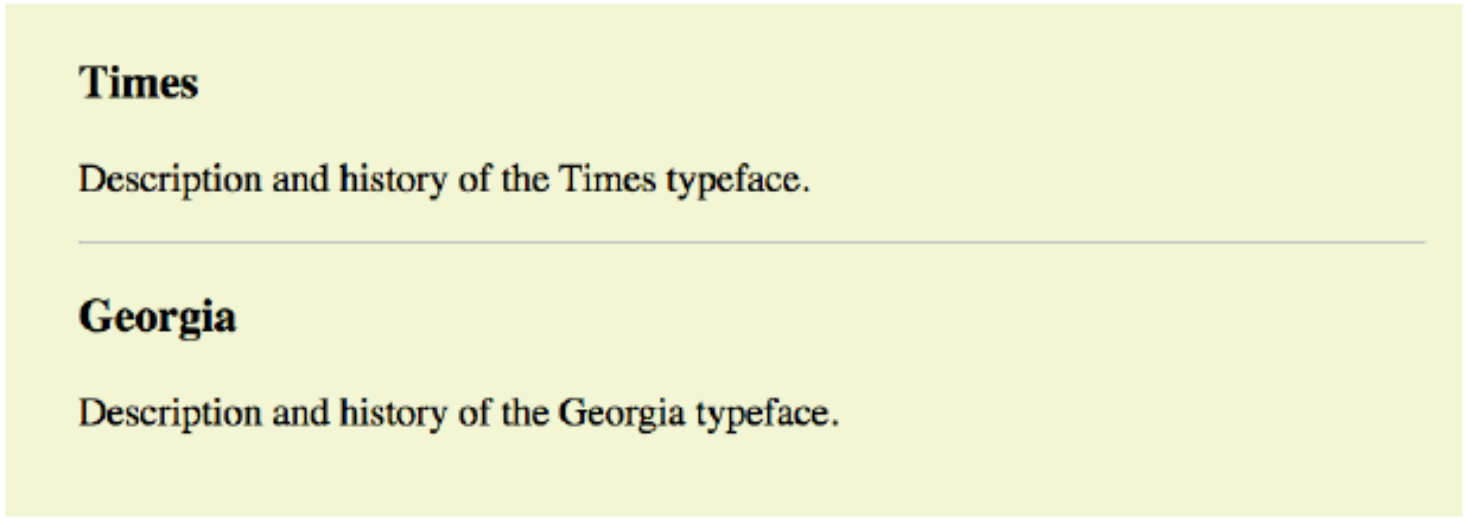


FIGURE 5-2. The default rendering of a thematic break (horizontal rule).

Thematic Break

Horizontal Ruler

```
<ul>
  <li>Serif</li>
  <li>Sans-serif</li>
  <li>Script</li>
  <li>Display</li>
  <li>Dingbats</li>
</ul>
```

FIGURE 5-3. The default rendering of the sample unordered list. The browser adds the bullets automatically.

Unordered List

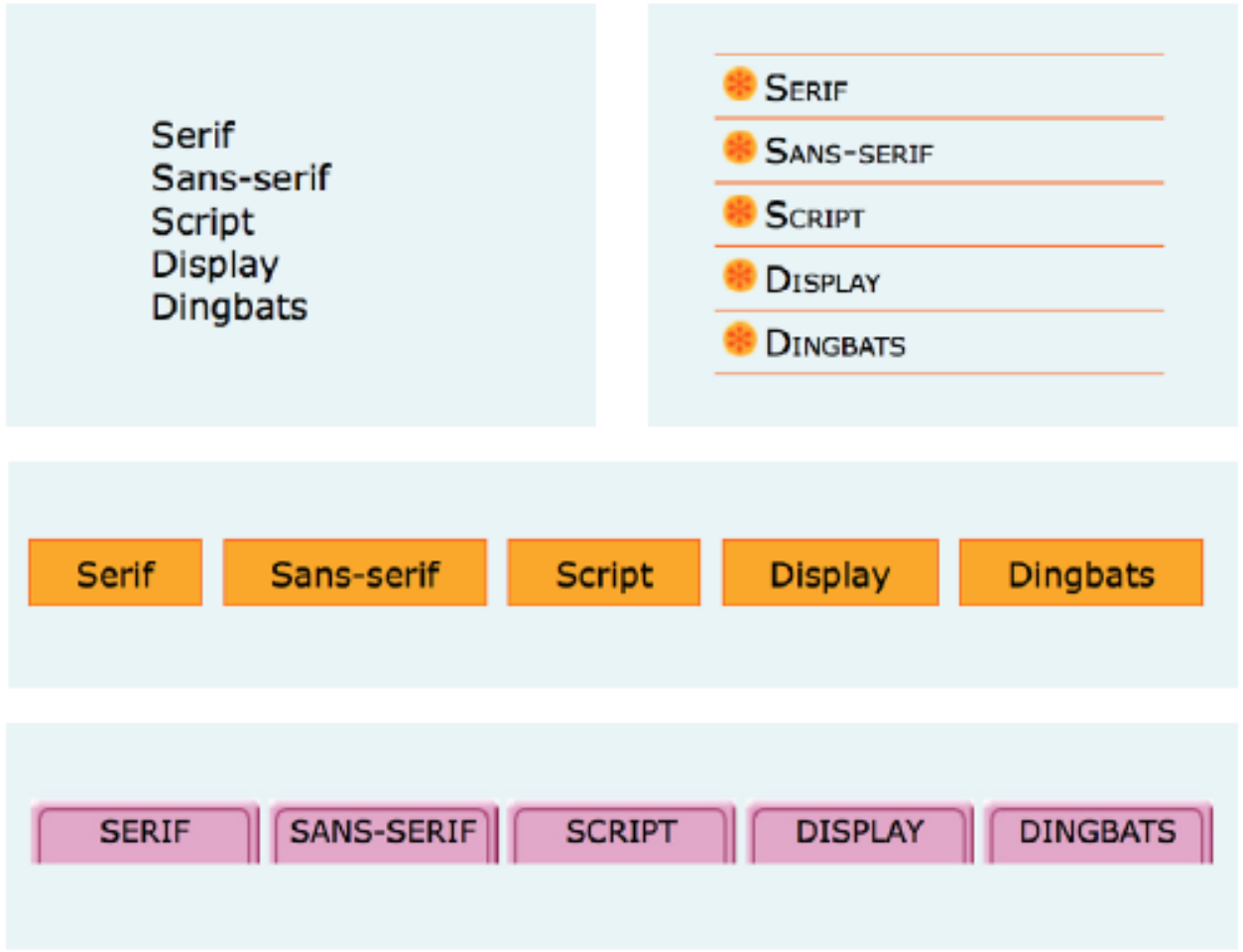


FIGURE 5-4. With style sheets, you can give the same unordered list many looks.

CSS-Styled Unordered List

```
<ol>
  <li>Gutenberg develops moveable type (1450s)</li>
  <li>Linotype is introduced (1890s)</li>
  <li>Photocomposition catches on (1950s)</li>
  <li>Type goes digital (1980s)</li>
</ol>
```

1. Gutenberg develops moveable type (1450s)
2. Linotype is introduced (1890s)
3. Photocomposition catches on (1950s)
4. Type goes digital (1980s)

FIGURE 5-5. The default rendering of an ordered list. The browser adds the numbers automatically.

Ordered List

```
<ol>
  <li>Gutenberg develops moveable type (1450s)</li>
  <li>Linotype is introduced (1890s)</li>
  <li>Photocomposition catches on (1950s)</li>
  <li>Type goes digital (1980s)</li>
</ol>
```

1. Gutenberg develops moveable type (1450s)
2. Linotype is introduced (1890s)
3. Photocomposition catches on (1950s)
4. Type goes digital (1980s)

FIGURE 5-5. The default rendering of an ordered list. The browser adds the numbers automatically.

Ordered List

```
<dl>
  <dt>Linotype</dt>
  <dd>Line-casting allowed type to be selected, used, then recirculated
into the machine automatically. This advance increased the speed of
typesetting and printing dramatically.</dd>

  <dt>Photocomposition</dt>
  <dd>Typefaces are stored on film then projected onto photo-sensitive
paper. Lenses adjust the size of the type.</dd>

  <dt>Digital type</dt>
  <dd><p>Digital typefaces store the outline of the font shape in a
format such as Postscript. The outline may be scaled to any size for
output.</p>
    <p>Postscript emerged as a standard due to its support of
graphics and its early support on the Macintosh computer and Apple
laser printer.</p>
  </dd>
</dl>
```

Description List I:

HTML

Linotype
: Line-casting allowed type to be selected, used, then recirculated into the machine automatically. This advance increased the speed of typesetting and printing dramatically.

Photocomposition
: Typefaces are stored on film then projected onto photo-sensitive paper. Lenses adjust the size of the type.

Digital type
: Digital typefaces store the outline of the font shape in a format such as Postscript. The outline may may be scaled to any size for output.

  Postscript emerged as a standard due to its support of graphics and its early support on the Macintosh computer and Apple laser printer.

FIGURE 5-6. The default rendering of a definition list. Definitions are set off from the terms by an indent.

Description List II:
Default Rendering

Text Content Tag (II)

Quotation:

<blockquote></blockquote>

/* text will be shown in quotation format */

Preformatted Text:

<pre></pre>

/* Text will shown as you type */

Figures:

<figure></figure>

/* include figure in text, normally image */

Figure Caption:

<figcaption> </figcaption>

/* description of the figure */

```
<p>Renowned type designer, Matthew Carter, has this to say about his
profession:</p>

<blockquote>
  <p>Our alphabet hasn't changed in eons; there isn't much latitude in
what a designer can do with the individual letters.</p>

  <p>Much like a piece of classical music, the score is written
down. It's not something that is tampered with, and yet, each
conductor interprets that score differently. There is tension in
the interpretation.</p>
</blockquote>
```

FIGURE 5-7 shows the default rendering of the **blockquote** example. This can be altered with CSS.

Renowned type designer, Matthew Carter, has this to say about his profession:

> Our alphabet hasn't changed in eons; there isn't much latitude in what a designer can do with the individual letters.
>
> Much like a piece of classical music, the score is written down. It's not something that is tampered with, and yet, each conductor interprets that score differently. There is tension in the interpretation.

FIGURE 5-7. The default rendering of a **blockquote** element.

Block Quote

```
<pre>
This is                 an              example of
      text with a           lot of
                            curious
                            whitespace.
</pre>

<p>
This is                 an              example of
      text with a           lot of
                            curious
                            whitespace.
</p>
```

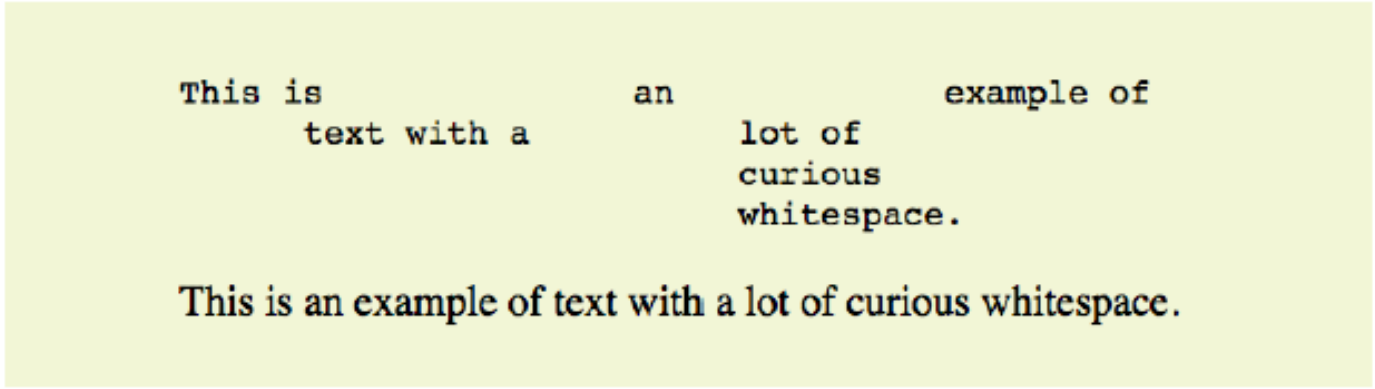


FIGURE 5-8. Preformatted text is unique in that the browser displays the whitespace exactly as it is typed into the source document. Compare it to the paragraph element, in which multiple line returns and character spaces are reduced to a single space.

Pre-formatted Text

```
<figure>
    <pre>
      <code>
    body {
      background-color: #000;
      color: red;
    }
       </code>
     </pre>
    <figcaption>Sample CSS rule.</figcaption>
</figure>
```

In EXERCISE 5-1, you'll get a chance to mark up a document yourself and try out the basic text elements we've covered so far.

Figures

Exercise 5-1

EXERCISE 5-1. Marking up a recipe

The owners of the Black Goose Bistro have decided to share recipes and news on their site. In the exercises in this chapter, we'll assist them with content markup.

In this exercise, you will find the raw text of a recipe. It's up to you to decide which element is the best semantic match for each chunk of content. You'll use **paragraphs**, **headings**, **lists**, and at least one **special content element**.

You can write the tags right on this page. Or, if you want to use a text editor and see the results in a browser, this text file, as well as the final version with markup, is available at *learningwebdesign.com/5e/materials*.

```
Tapenade (Olive Spread)

This is a really simple dish to prepare and it's always a big hit at
parties. My father recommends:

"Make this the night before so that the flavors have time to blend. Just
bring it up to room temperature before you serve it. In the winter, try
serving it warm."

Ingredients

1 8oz. jar sundried tomatoes
2 large garlic cloves
2/3 c. kalamata olives
1 t. capers

Instructions

Combine tomatoes and garlic in a food processor. Blend until as smooth
as possible.

Add capers and olives. Pulse the motor a few times until they are
incorporated, but still retain some
texture.

Serve on thin toast rounds with goat cheese and fresh basil garnish
(optional).
```

Special Text blocks (I)

Text division: /* division for html 4 */

Span: /* to define a span of text for special format or link */

 /* text in ***emphasized*** format */

<small></small> /* text in small letter small */

Subscripts: /* text in subscripts CO_2 */

Superscripts: /* text in superscripts x^2 */

Strike through: <s></s> /* text in ~~strike through~~ format */

Bold: ; Italics: ; underline: <u></u>; **emphasized text:<**em>; short quotation: <q></q>

Hyperlink: <a> /* define a link to another page */

: line break; <wbr>: word break; <abbr></abbr>: abbreviation;

Special character: , &amp /* see table 5-2 for details */

Special Text Blocks (II):

Text in special style

Machine Readable Data: <data></data>

Inserted text: <ins></ins>; deleted text: <del></del> /* put text in different style so people know the text is either inserted or deleted in a new version */

Defining Term:<dfn></dfn> /* mark text as definition style */

Citation: <cite></cite>

Program code: <code></code>

Variable: <var></var>

Program sample: <samp></samp>

User-entered keyboard strokes: <kbd></kbd>

TABLE 5-1. Text-level semantic elements

Element	Description
a	An anchor or hypertext link (see **Chapter 6** for details)
abbr	Abbreviation
b	Added visual attention, such as keywords (bold)
bdi	Indicates text that may have directional requirements
bdo	Bidirectional override; explicitly indicates text direction (left to right, **ltr**, or right to left, **rtl**)
br	Line break
cite	Citation; a reference to the title of a work, such as a book title
code	Computer code sample
data	Machine-readable equivalent dates, time, weights, and other measurable values
del	Deleted text; indicates an edit made to a document
dfn	The defining instance or first occurrence of a term
em	Emphasized text
i	Alternative voice (italic) or alternate language
ins	Inserted text; indicates an insertion in a document
kbd	Keyboard; text entered by a user (for technical documents)
mark	Contextually relevant text
q	Short, inline quotation
ruby, rt, rp	Provides annotations or pronunciation guides under East Asian typography and ideographs
s	Incorrect text (strike-through)
samp	Sample output from programs
small	Small print, such as a copyright or legal notice (displayed in a smaller type size)
span	Generic phrase content
strong	Content of strong importance
sub	Subscript
sup	Superscript
time	Machine-readable time data
u	Indicates a formal name, misspelled word, or text that would be underlined
var	A variable or program argument (for technical documents)
wbr	Word break

Text-Level (Inline) Elements

b — The slabs at the ends of letter strokes are called **serifs**.

i — Simply change the font and *Voila!*, a new personality!

s — Scala Sans was designed by ~~Eric Gill~~ Martin Majoor.

u — New York subway signage is set in Helviteca.

small — (This font is free for personal and commercial use.)

FIGURE 5-10. The default rendering of `b`, `i`, `s`, `u`, and `small` elements.

```
<p>H<sub>2</sub>O</p>

<p>E=MC<sup>2</sup></p>
```

H_20 $E=MC^2$

FIGURE 5-12. Subscript and superscript

Text-Level Elements

Elements originally named for their presentational properties

```
<p>So much depends <br>upon <br><br>a red wheel <br>barrow</p>
```

So much depends
upon

a red wheel
barrow

FIGURE 5-15. Line breaks are inserted at each **br** element. (Example extracted from "The Red Wheelbarrow" by William Carlos Williams.)

```
<p>The biggest word you've ever heard and this is how it goes:
<em>supercali<wbr>fragilistic<wbr>expialidocious</em>!</p>
```

The biggest word you've ever heard and this is how it goes: *supercalifragilistic expialidocious*!

FIGURE 5-16. When there is not enough room for a word to fit on a line, it will break at the location of the **wbr** element.

Breaks

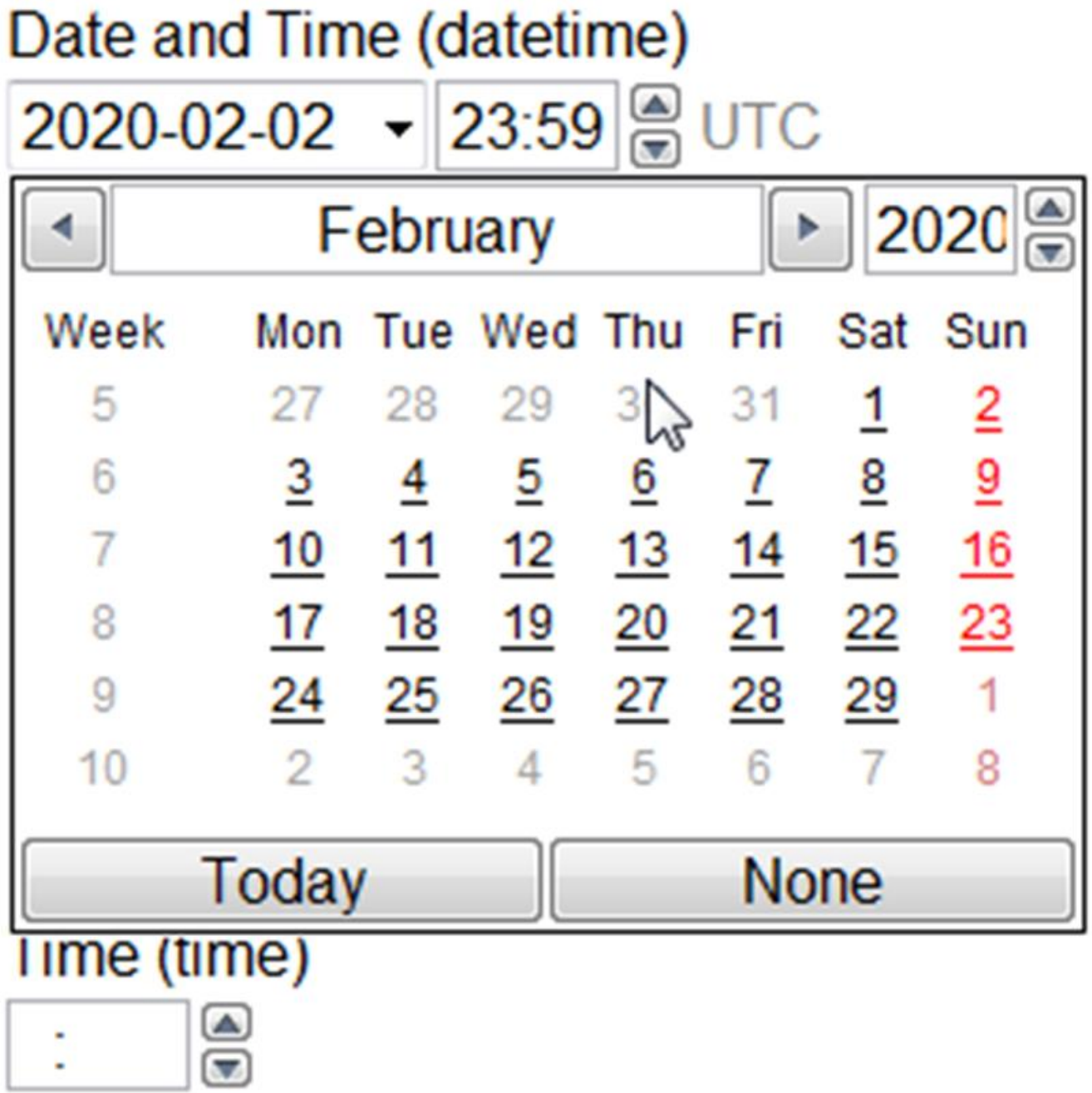
Date and Time special tag

Time data:

<time datetime="2012-09-01T20:00-05:00"></time>

/* this is text format */

Compared with Date and Time input form in html



Here are a few examples of valid values for **datetime**:

- **Time only:** 9:30 p.m.

```
<time datetime="21:30">9:30p.m.</time>
```

- **Date only:** June 19, 2016

```
<time datetime="2016-06-19">June 19, 2016</time>
```

- **Date and time:** Sept. 5, 1970, 1:11a.m.

```
<time datetime="1970-09-05T01:11:00">Sept. 5, 1970, 1:11a.m.</time>
```

- **Date and time, with time zone information:** 8:00am on July 19, 2015, in Providence, RI

```
<time datetime="2015-07-19T08:00:00-05:00">July 19, 2015, 8am,
Providence RI</time>
```

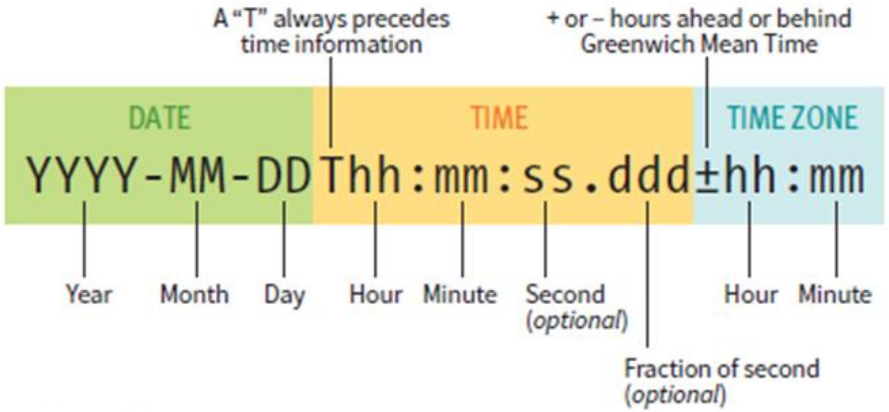


FIGURE 5-14. *Standardized date and time syntax.*

Standard Text and Date Elements

Tags for Essay or Reports

Header: <header>

Division in articles: <section>

Article: <article>

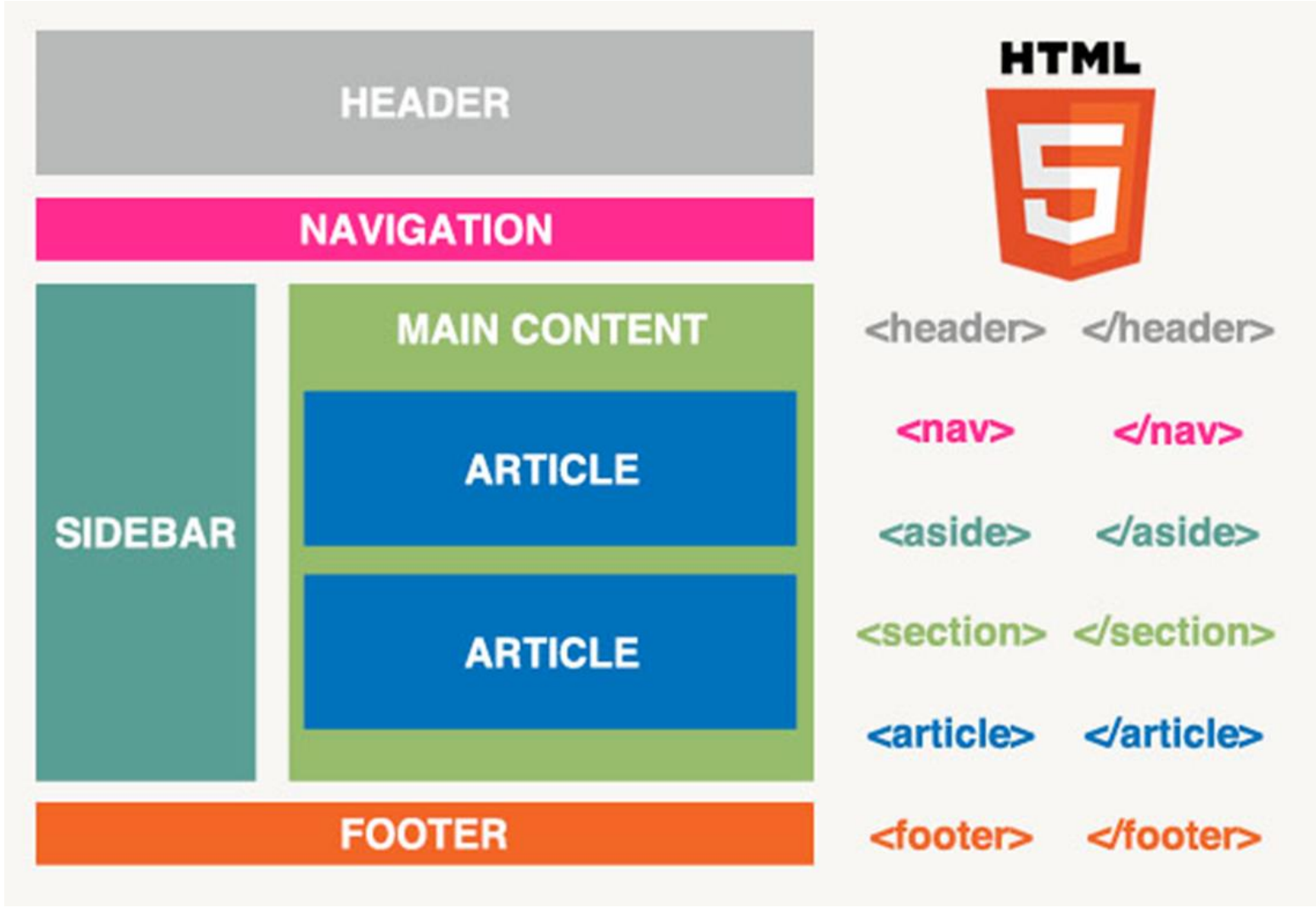
Navigation Link Bar: <nav>

Side notes: <aside>

Foot notes: <footer>

Copyright words and address: <address>

Note: These divisions are used to provide special CSS style. Otherwise HTML 4's division is good enough to provide structural information



```
<body>
<header>…</header>
<main>
   <h1>Humanist Sans Serif</h1>
   <!-- code continues -->
</main>
</body>
```

Main Content

<main></main>

Primary content area of page or app

```
<body>
<header>
  <img src="/images/logo.png" alt="logo">
  <h1>Nuts about Web Fonts</h1>
  <nav>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/">Blog</a></li>
      <li><a href="/">Shop</a></li>
    </ul>
  </nav>
</header>
<!--page content-->
</body>
```

Headers and Footers

<header>…</header>
Introductory material for page, section, or article

```
<article>
  <header>
    <h1>More about WOFF</h1>
    <p>by Jennifer Robbins, <time datetime="2017-11-11">November 11,
2017</time></p>
  </header>
  <!-- article content here -->
  <footer>
    <p><small>Copyright &copy;2017 Jennifer Robbins.</small></p>
    <nav>
      <ul>
        <li><a href="/">Previous</a></li>
        <li><a href="/">Next</a></li>
      </ul>
    </nav>
  </footer>
</article>
```

Headers and Footers

<footer>…</footer>

Footer for page, section, or article

```
<section>
  <h2>Typography Books</h2>
  <ul>
     <li>…</li>
  </ul>
</section>

<section>
  <h2>Online Tutorials</h2>
  <p>These are the best tutorials on the web.</p>
  <ul>
    <li>…</li>
  </ul>
</section>
```

Sections and Articles

<section>…</section>

Thematic group of content

```
<article>
  <h1>Get to Know Helvetica</h1>
  <section>
    <h2>History of Helvetica</h2>
    <p>…</p>
  </section>
  <section>
    <h2>Helvetica Today</h2>
    <p>…</p>
  </section>
</article>
```

Sections and Articles

<article>…</article>

Self-contained, reusable composition

```
<section id="essays">
  <article>
    <h1>A Fresh Look at Futura</h1>
    <p>…</p>
  </article>

  <article>
    <h1>Getting Personal with Humanist</h1>
    <p>…</p>
  </article>
</section>
```

Sections and Articles

Article in sections

```
<h1>Web Typography</h1>
<p>Back in 1997, there were competing font formats and tools for
making them…</p>
<p>We now have a number of methods for using beautiful fonts on web
pages…</p>
<aside>
  <h2>Web Font Resources</h2>
  <ul>
    <li><a href="http://typekit.com/">Typekit</a></li>
    <li><a href="http://fonts.google.com">Google Fonts</a></li>
  </ul>
</aside>
```

Aside (Sidebars)

<aside>…</aside>

Tangentially related material

```
<nav>
  <ul>
    <li><a href="/">Serif</a></li>
    <li><a href="/">Sans-serif</a></li>
    <li><a href="/">Script</a></li>
    <li><a href="/">Display</a></li>
    <li><a href="/">Dingbats</a></li>
  </ul>
</nav>
```

Navigation

<nav>…</nav>

Primary navigation links

```
<address>
Contributed by <a href="../authors/robbins/">Jennifer Robbins</a>,
<a href="http://www.oreilly.com/">O'Reilly Media</a>
</address>
```

Addresses

<address>…</address>

Contact information

```
<div class="listing">
  <img src="images/felici-cover.gif" alt="">
  <p><cite>The Complete Manual of Typography</cite>, James Felici</p>
  <p>A combination of type history and examples of good and bad type
design.</p>
</div>
```

```
<ul>
  <li>John: <span class="tel">999.8282</span></li>
  <li>Paul: <span class="tel">888.4889</span></li>
  <li>George: <span class="tel">888.1628</span></li>
  <li>Ringo: <span class="tel">999.3220</span></li>
</ul>
```

Panacea

Medicine for All

** … **

Generic block-level element

…

Generic inline element

Character Escapes

- There are two ways of referring to (escaping) a specific character:
  - Using a predefined abbreviated name for the character (called a named entity; see Note).
  - Using an assigned numeric value that corresponds to its position in a coded character set (numeric entity). Numeric values may be in decimal or hexadecimal format.
- All character references begin with an **&** (ampersand) and end with a **;** (semicolon).

Element Attributes

SECTION 3

Properties of an Element as an identifier

- **id :**
- store is the identifier for the division. id can be used to identify any individual element.

Properties of an Element as an identifier

- **class:**
- There might be many division sharing the same class so that they can share some same features such as same background color or something.
- It can be viewed as a group id. This class group can share same CSS style

Properties of an Element as an identifier

- **name:**
- name is used as variable name for an element. For example, an input form can have a variable named xxx, when the placeholder is assigned value of “bbb”, then aaa = “bbb”
  - <input name=“comment” placeholder=“This is one.”> Then name is assigned a value of “This is one.” name can be viewed as another way of grouping a few input form as one same type to assign value to a same variable.
  - <input type=“radio” name=“gender” value=“boy”>
  - <input type=“radio” name=“gender” value=“girl”>

Properties of an Element as an identifier

- **value:**
- default value for an input form.

Adding Links

SECTION 4

SEO Cheat Sheet: Anatomy of A URL

seomoz.org
Read SEOmoz. Rank Better

1 SEO-FRIENDLY URL

1. Protocol
2. Subdomain
3. Domain
4. Top-Level Domain
5. Folders / Paths
6. Page
7. Named Anchor

Keyword Priority[1]
Observed Google priority of keyword placement:

(1) Domain
(2) Subdomain
(3) Folder
(4) Path/Page

SEO Tips for URLs

- Use subdomains carefully. They may be treated as separate entities, splitting domain authority.
- Separate path & page keywords with hyphens ("-").
- Anchors may help engines understand page structure.
- Keyword effectiveness in URLs decreases as URL length and keyword position increases.[1]

[1] SEOmoz correlational data (2009)

2 OLD DYNAMIC URL

1. Protocol
2. Subdomain
3. Domain
4. Top-Level Domain
5. Page / File Name
6. File Extension
7. CGI Parameters

Popular TLDs[2]

.com - commercial
.net - infrastructure
.org - non-profit
.edu - schools
.info - informational
.biz - small business
.name - personal sites

Popular ccTLDs*

.cn - China
.de - Germany
.uk - United Kingdom
.nl - Netherlands
.eu - European Union
.ru - Russian Federation
.ar - Argentina

Popular Extensions

.htm - Static HTML
.html - Static HTML
.php - PHP code
.asp - ASP code
.aspx - ASP.NET
.cfm - ColdFusion
.jsp - Java Code

[2] Verisign domain report (2009)

* ccTLD = Country Code TLD

Anchor Tag <a>

- Anchor tag is used to anchor any URL link into the html document.
- It can be used to locate a web page, an image, a text file, a video file and any other file type which can be found by the URL provided
- Any URL with #fragment can be used to locate any element in a html. The #fragment is used to identify the element with id="fragment",
- The URL is specified by the href attribute (hypertext reference)
- The text written in between <a> and is the link access point on the html page, which means if the text is clicked, the target resource will be accessed.

The HREF Attribute

- You'll need to tell the browser which document to link to, right? The href (hypertext reference) attribute provides the address of the page or resource (its URL) to the browser. The URL must always appear in quotation marks.
- Most of the time you'll point to other HTML documents; however, you can also point to other web resources, such as images, audio, and video files.
- Because there's not much to slapping anchor tags around some content, the real trick to linking comes in getting the URL correct.

Absolute URLs

- Absolute URLs provide the full URL for the document, including the protocol (**http://** or **https://**), the domain name, and the pathname as necessary. You need to use an absolute URL when pointing to a document out on the web (i.e., not on your own server):
- href="http://www.oreilly.com/"
- Sometimes, when the page you're linking to has a long URL pathname, the link can end up looking pretty confusing (FIGURE 6-2). Just keep in mind that the structure is still a simple container element with one attribute.
- Don't let the long pathname intimidate you.

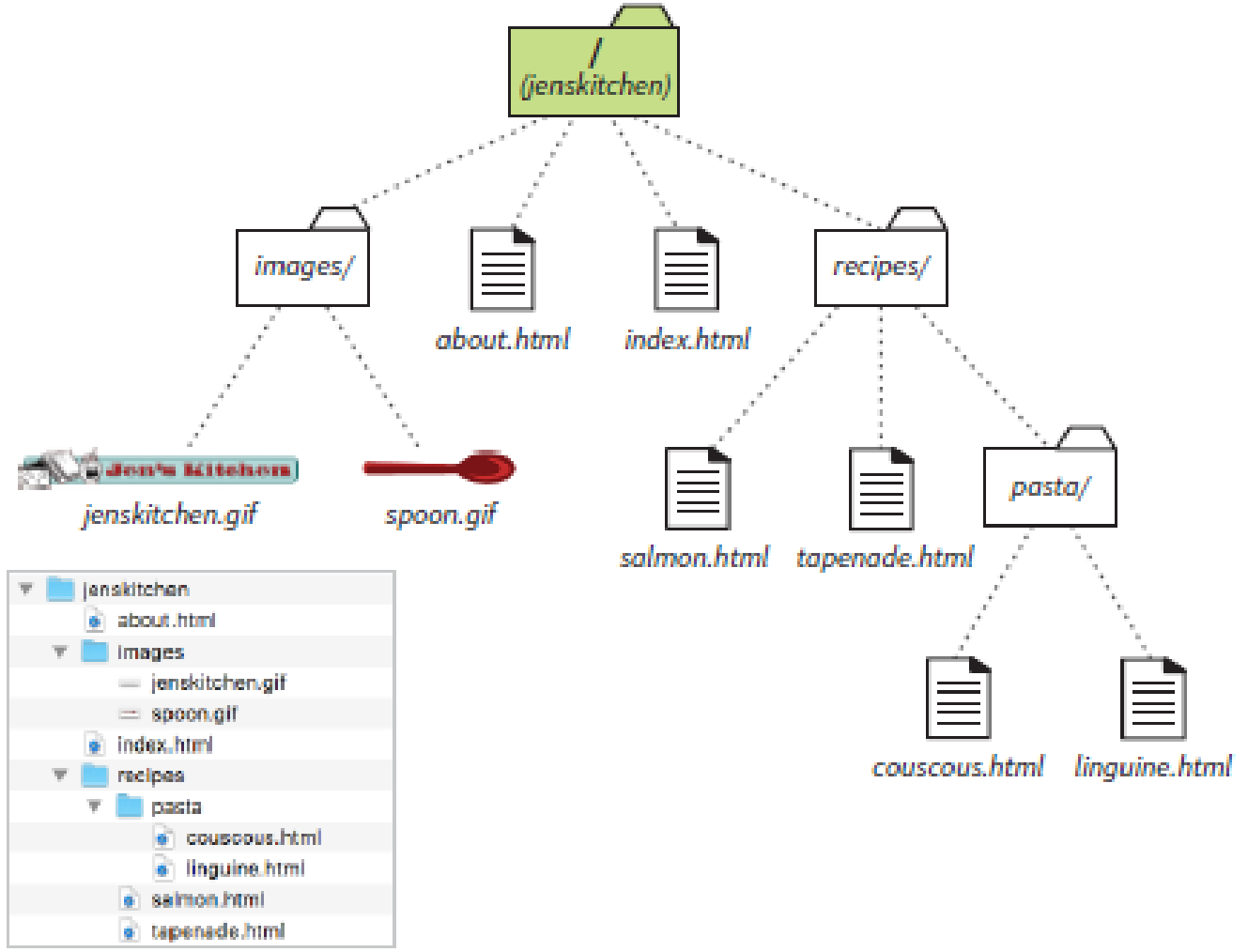
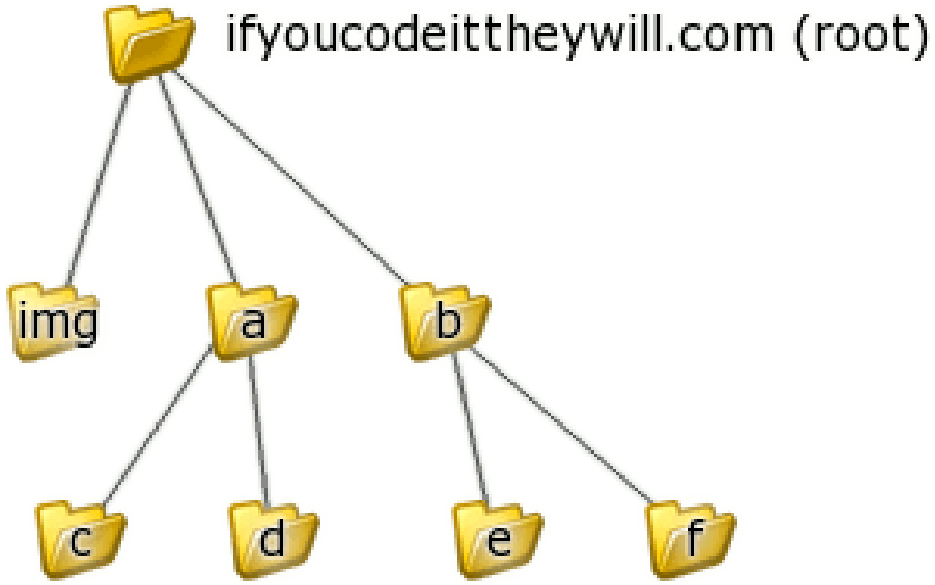


FIGURE 6-4. A diagram of the *jenskitchen* site structure.

Relative URL



- A web-site (ifyoucodeithteywill.com) has a root directory as shown on the left.
- URL http://www.ifyoucodetheywill.com will access index.htm or index.html at root directory
- In index.htm file, URL of a/c/test.htm will access the test.htm file in c directory
- .. means upper level directory
- . Means current directory
- For test.htm, ../../b/e/test2.htm will access test3.htm in e directory

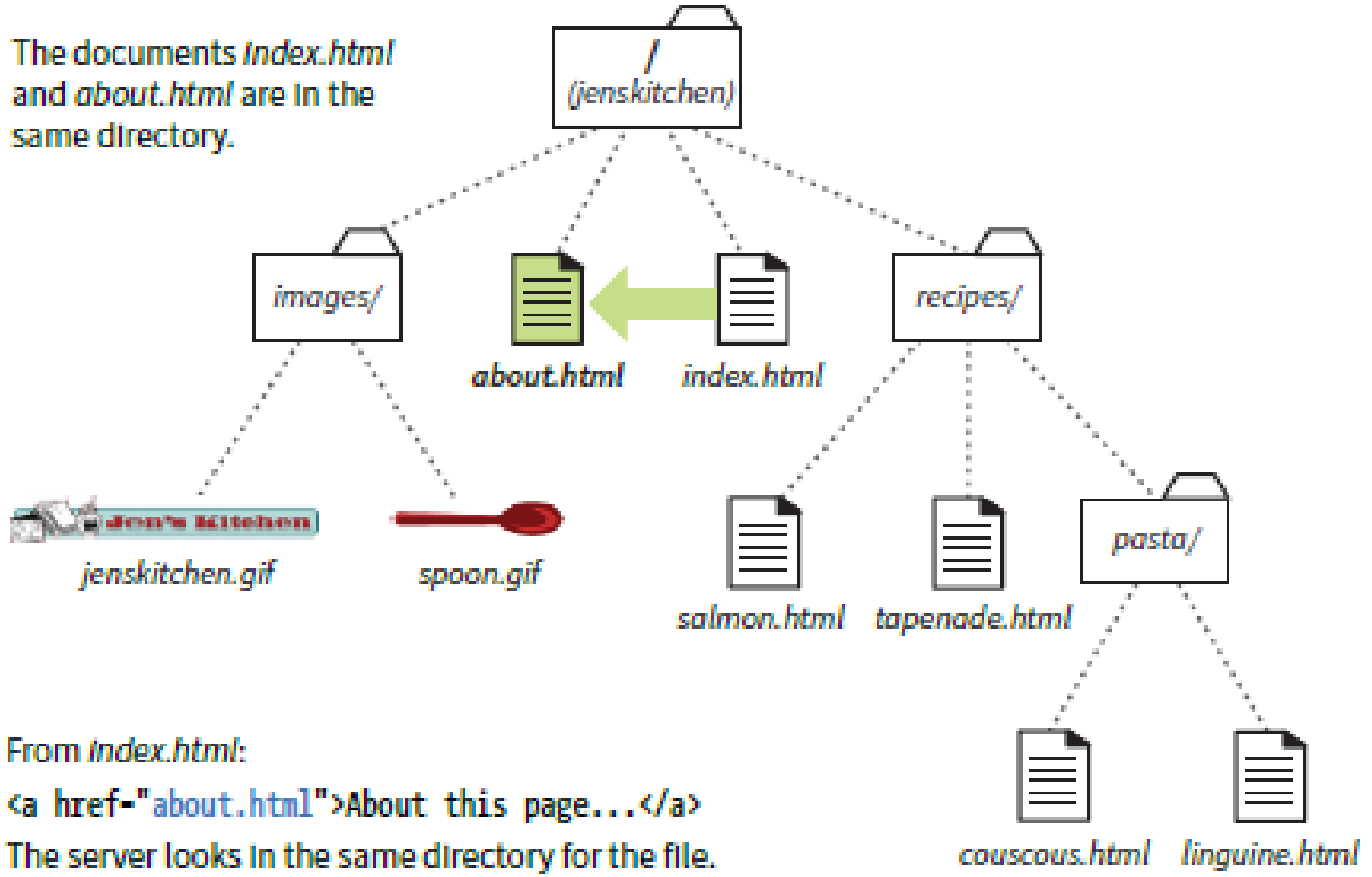


FIGURE 6-5. Writing a relative URL to another document in the same directory.

```
<li><a href="recipes/salmon.html">Garlic Salmon</a></li>
```

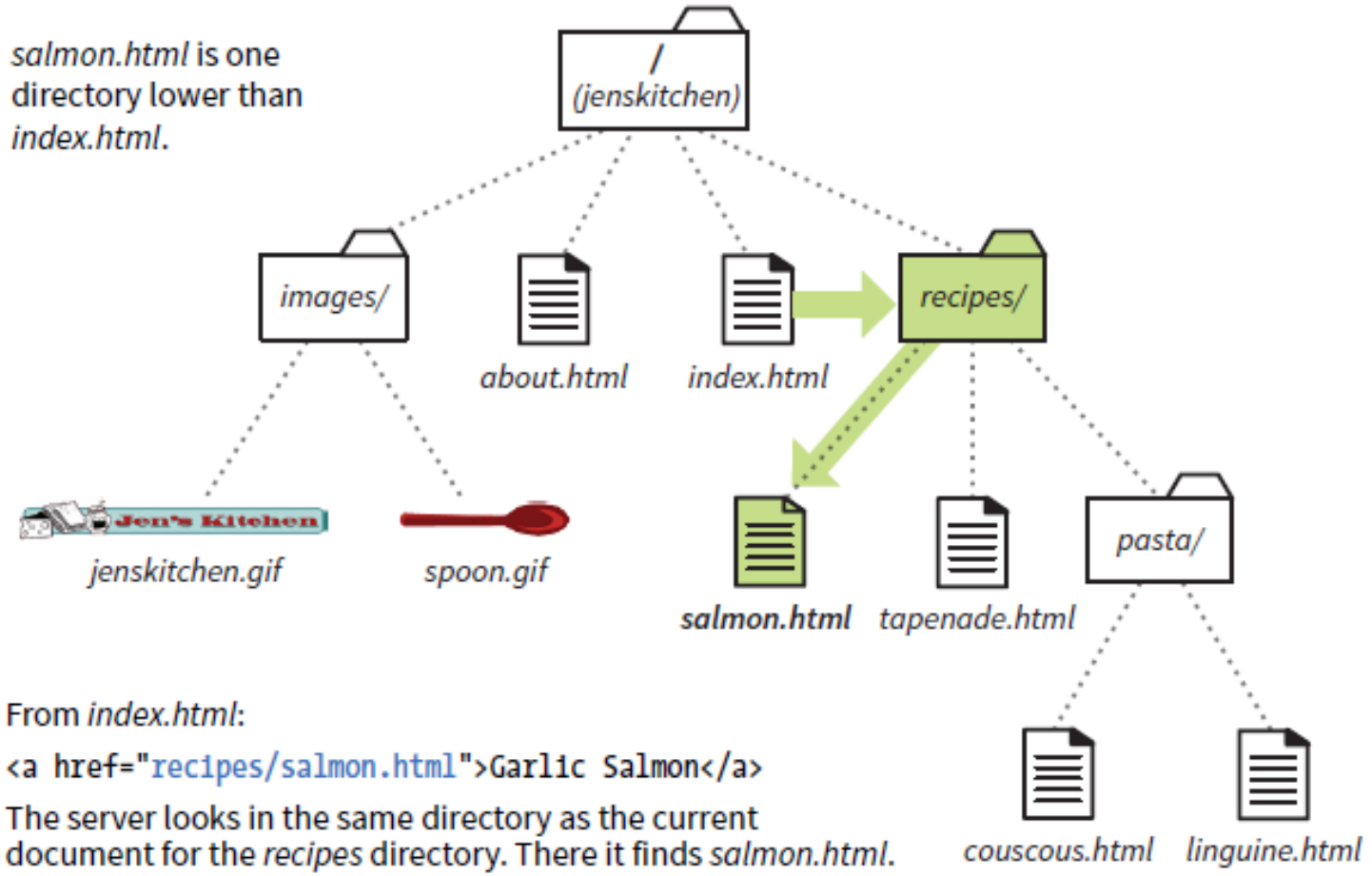


FIGURE 6-6. Writing a relative URL to a document that is one directory level lower than the current document.

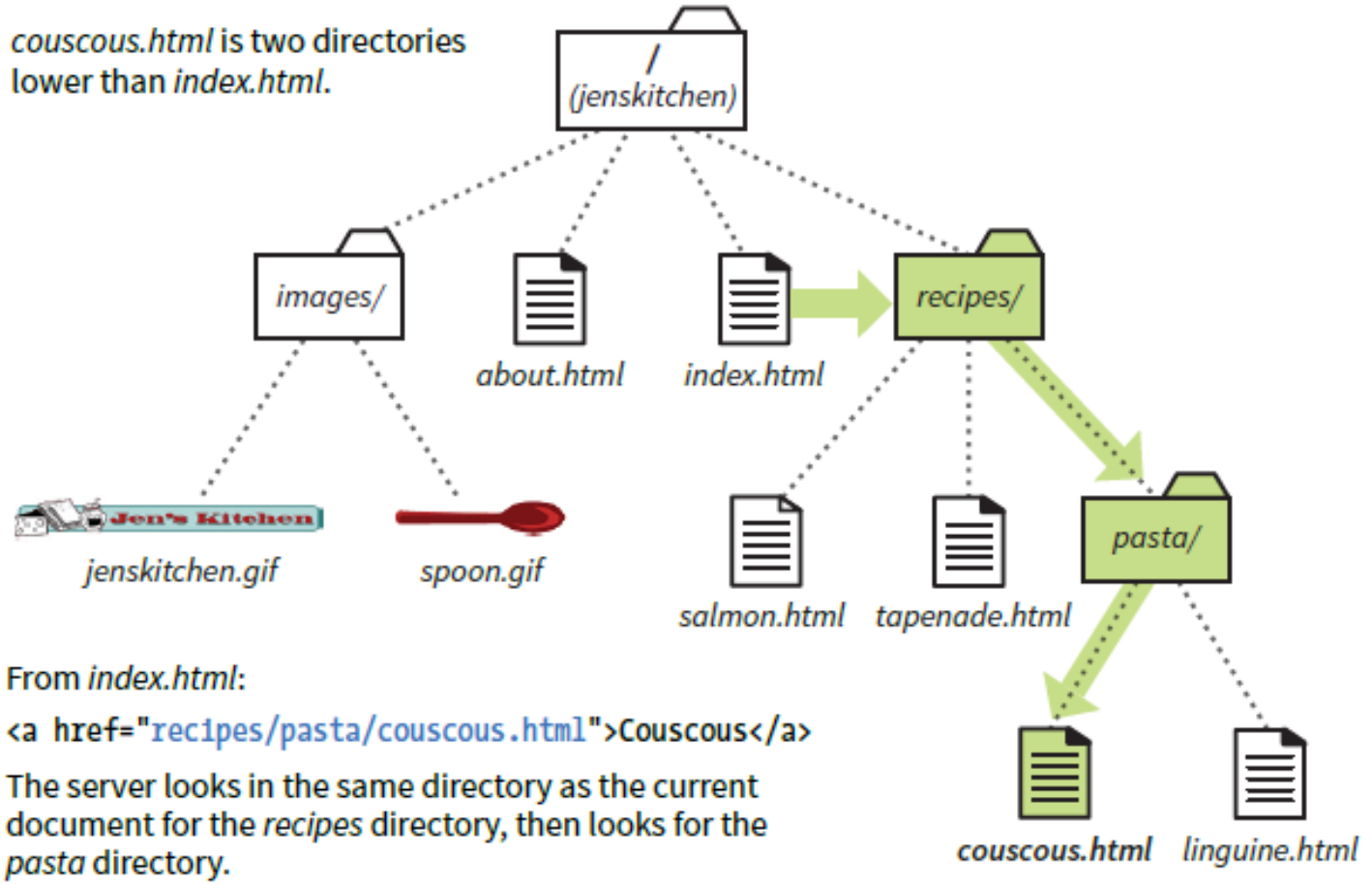


FIGURE 6-7. Writing a relative URL to a document that is two directory levels lower than the current document. You can try it yourself in EXERCISE 6-4.

```
<p><a href="../index.html">[Back to home page]</a></p>
```

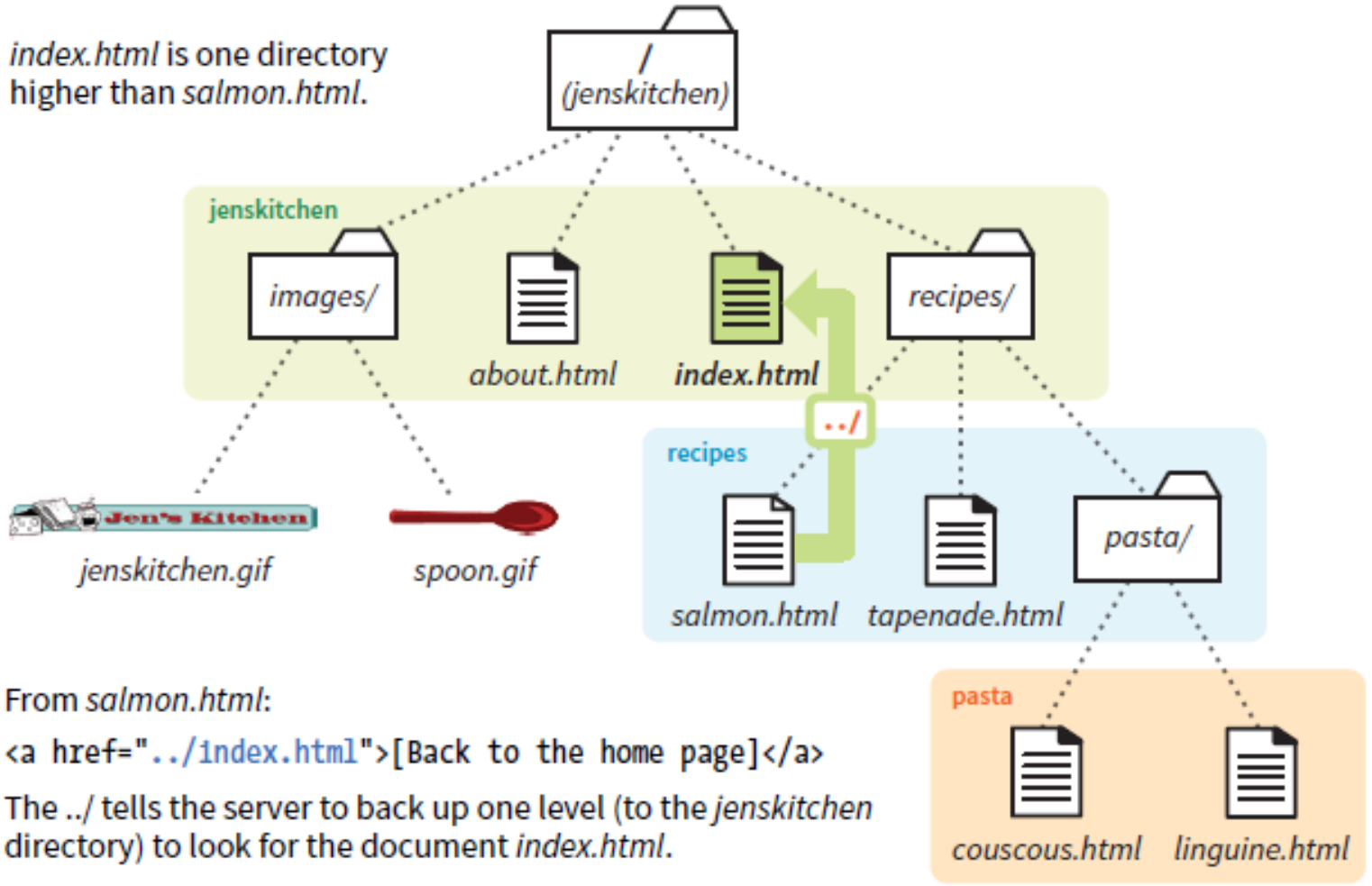


FIGURE 6-8. Writing a relative URL to a document that is one directory level higher than the current document.

```
<p><a href="../../index.html">[Back to home page]</a></p>
```

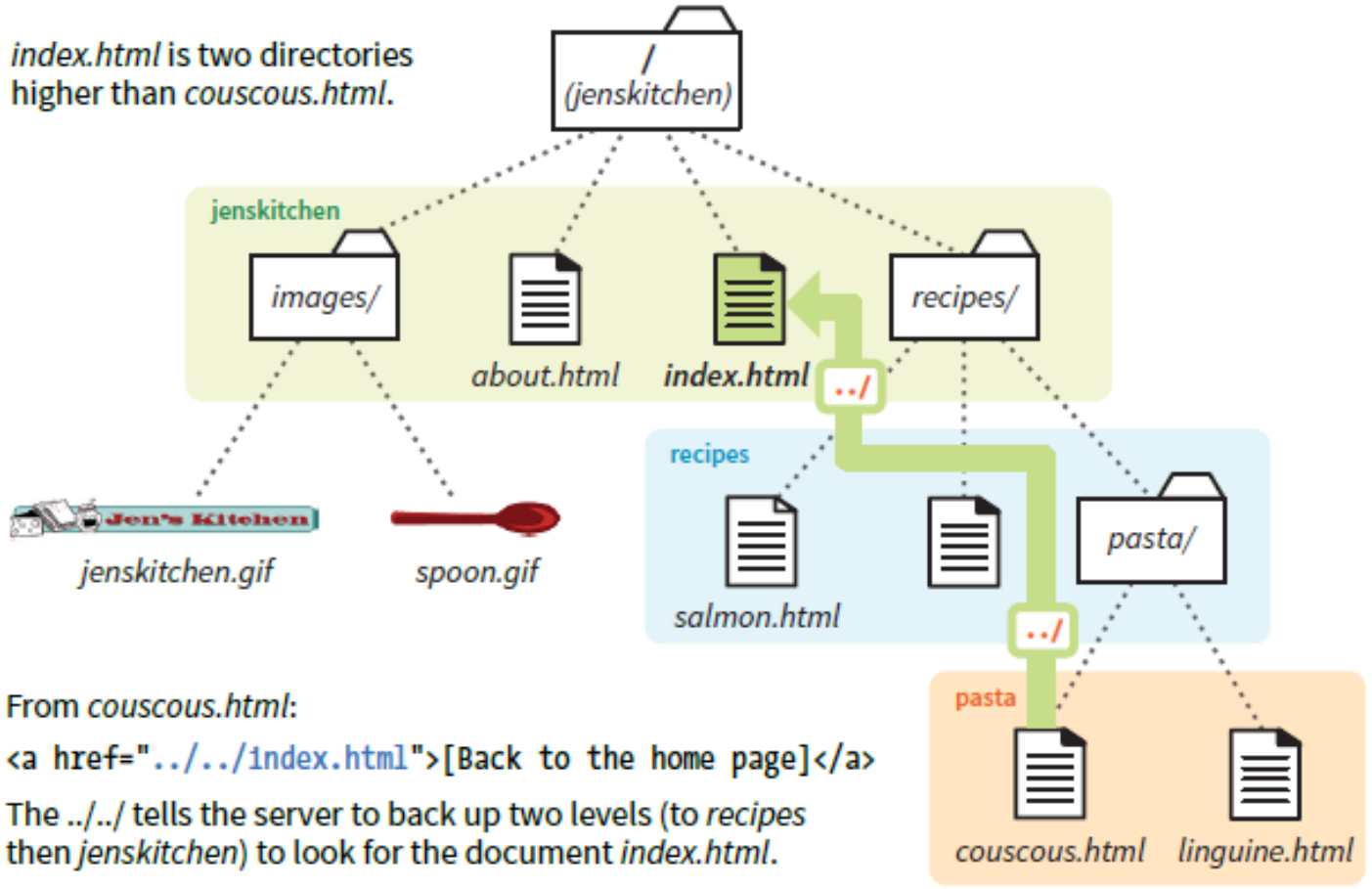


FIGURE 6-9. Writing a relative URL to a document that is two directory levels higher than the current document.

```
<a href="/recipes/salmon.html">Garlic Salmon</a>
```

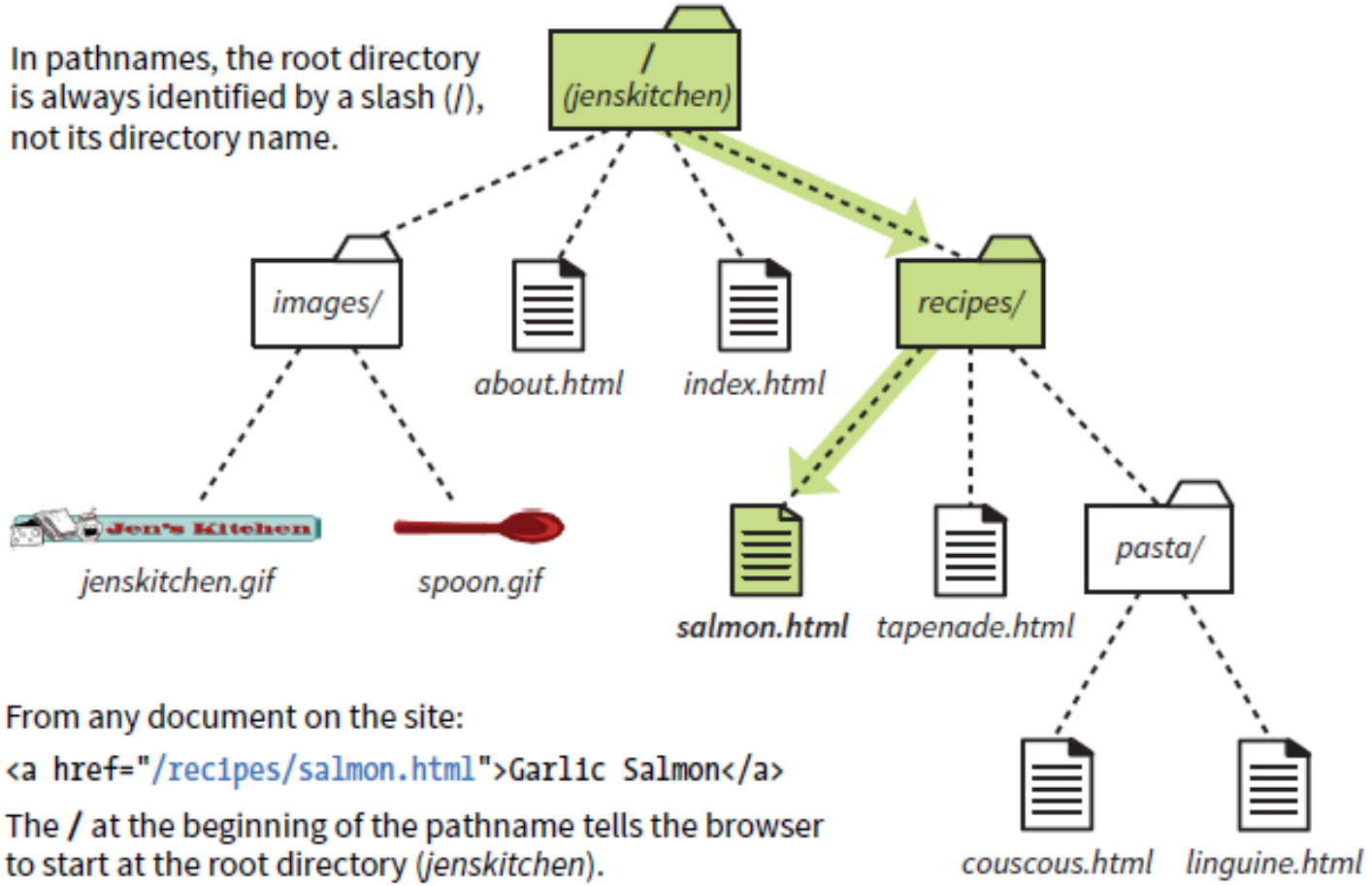


FIGURE 6-10. Writing a relative URL starting at the root directory.

Writing Pathnames to Images

- The src attribute in the img element works the same as the href attribute in anchors. Because you'll most likely be using images from your own server, the src attributes within your image elements will be set to relative URLs.
- Let's look at a few examples from the Jen's Kitchen site. First, to add an image to the index.html page, you'd use the following markup:

```
<img src="images/jenskitchen.gif" alt="">
```

- The URL says, "Look in the current directory (jenskitchen) for the images directory; in there you will find jenskitchen.gif."
- Now for the pièce de résistance. Let's add an image to the file couscous.html:

```
<img src="../../images/spoon.gif" alt="">
```

Writing Pathnames to Images

- This is a little more complicated than what we've seen so far. This pathname tells the browser to go up two directory levels to the top-level directory and, once there, look in the images directory for an image called spoon.gif. Whew!
- Of course, you could simplify that path by going the site root relative route, in which case the pathname to spoon.gif (and any other file in the images directory) could be accessed like this:

```
<img src="/images/spoon.gif" alt="">
```

- The trade-off is that you won't see the image in place until the site is uploaded to the server, but it does make maintenance easier once it's there.

Linking to a Specific Point in a Page

Step 1: Identifying the destination

Step 2: Linking to the destination

1. Identify the destination by using the `id` attribute.

```
<h2 id="startH">H</h2>
<dl>
<dt>hexadecimal</dt>
...
```

2. Create a link to the destination. The # before the name is necessary to identify this as a fragment and not a filename.

```
<p>... | F | G | <a href="#startH">H</a> | I | J ...</p>
```

Linking to a fragment

Step 1: Identifying the destination

Step 2: Linking to the destination

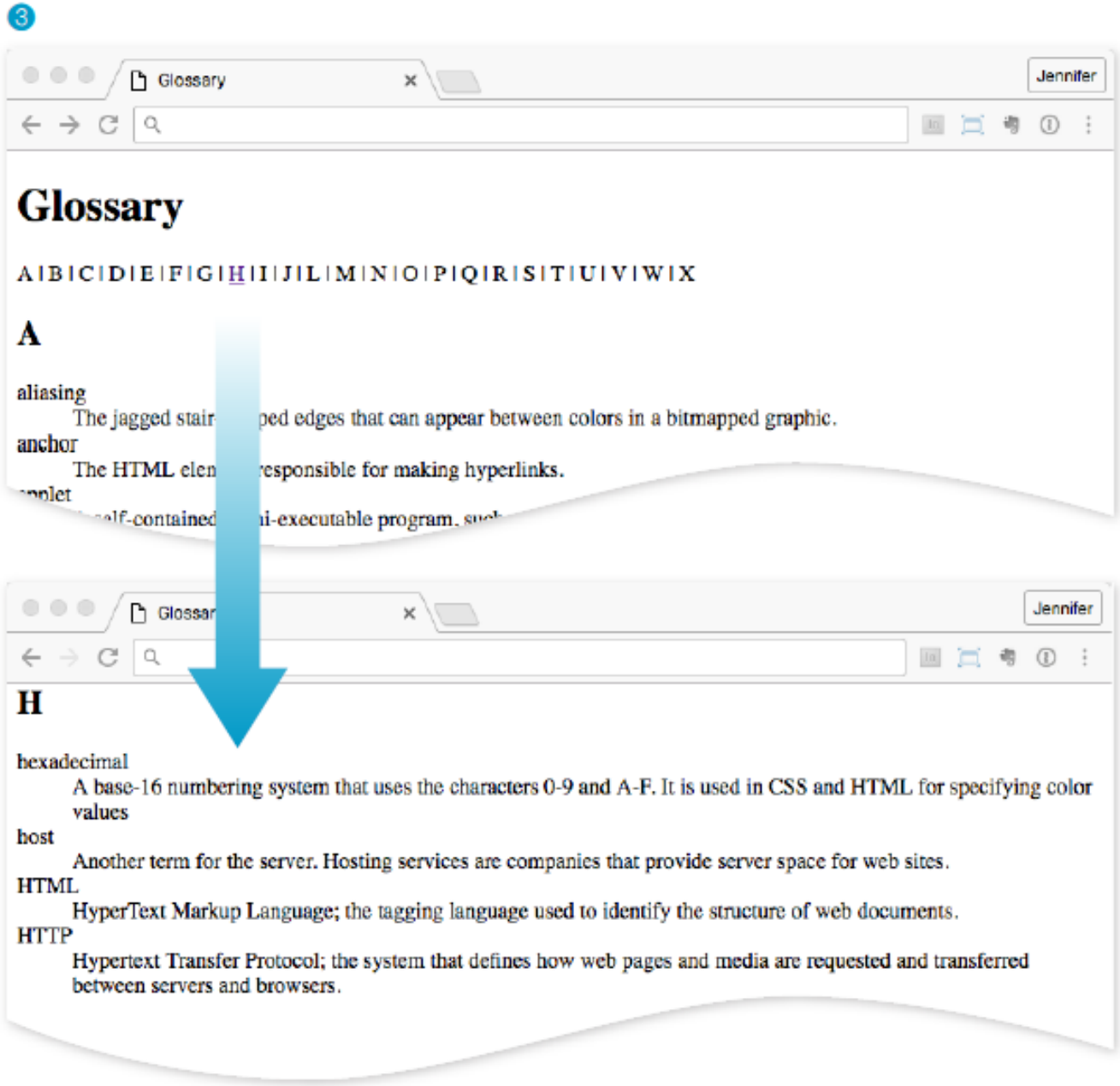


FIGURE 6-11. Linking to a specific destination (a fragment) within a single web page.

Linking to a Fragment in Another Document

See the Glossary, letter H

See the Glossary, letter H

Targeting a New Browser Window

- To open a new window with markup, use the target attribute in the anchor (a) element to tell the browser the name of the window in which you want the linked document to open. Set the value of target to **_blank** or to any name of your choosing.

- Remember that with this method, you have no control over the size of the window, but it will generally open as a new tab or in a new window the same size as the most recently opened window in the user's browser. The new window may or may not be brought to the front depending on the browser and device used.

Targeting a New Browser Window

- Setting target="_blank" always causes the browser to open a fresh window.
- For example:

```
<a href="http://www.oreilly.com"
target="_blank">O'Reilly</a>
```

- If you include target="_blank" for every link, every link will launch a new window, potentially leaving your user with a mess of open windows. There's nothing wrong with it, per se, as long as it is not overused.

Mail Link and Telephone Link

- A sample **mailto** link is shown here:

```
<a href="mailto:alklecker@example.com">Contact Al Klecker</a>
```

- As you can see, it's a standard anchor element with the **href** attribute. But the value is set to **mailto:***name@address.com*.

- The syntax uses the **tel:** protocol and is very simple:

```
<a href="tel:+01-800-555-1212">Call us free at (800) 555-1212</a>
```

Image Tags

SECTION 5

```
<p>This summer, try making pizza <img src="pizza.png" alt="">
 on your grill.</p>
```

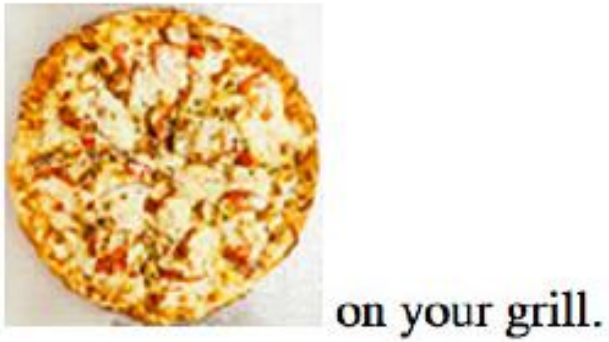


FIGURE 7-2. By default, images are aligned with the baseline of the surrounding text and do not cause a line break.

The img tag

<img>

Adds an inline image

Providing the Location with src

```
<img src="icon.gif" alt="">

<img src="/images/arrow.gif" alt="">

<img src="http://www.example.com/images/smile.gif"
     alt="">
```

Providing Alternative Text with alt

alt="*text*"

Alternative text

```
<a href="application.pdf">High school application</a>

<img src="images/pdflogo.png
     alt="PDF file">
```

Providing the Dimensions with width and height

width="*number*"
Image width in pixels

height="*number*"
Image height in pixels

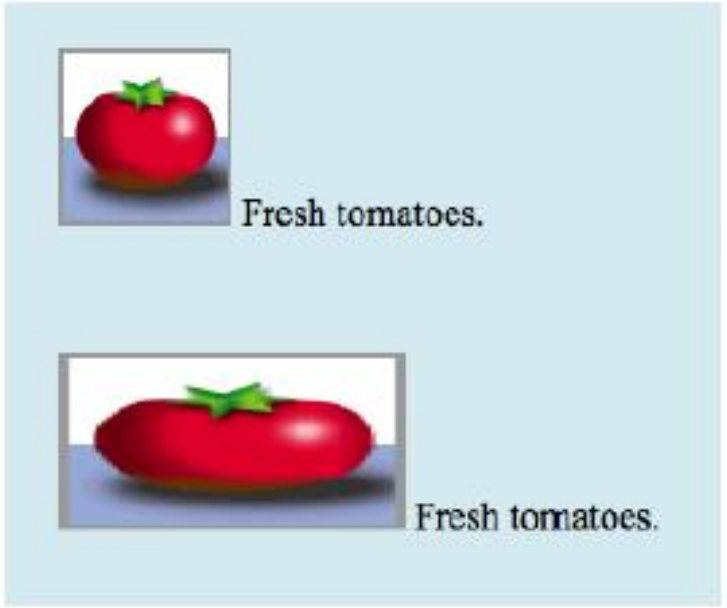


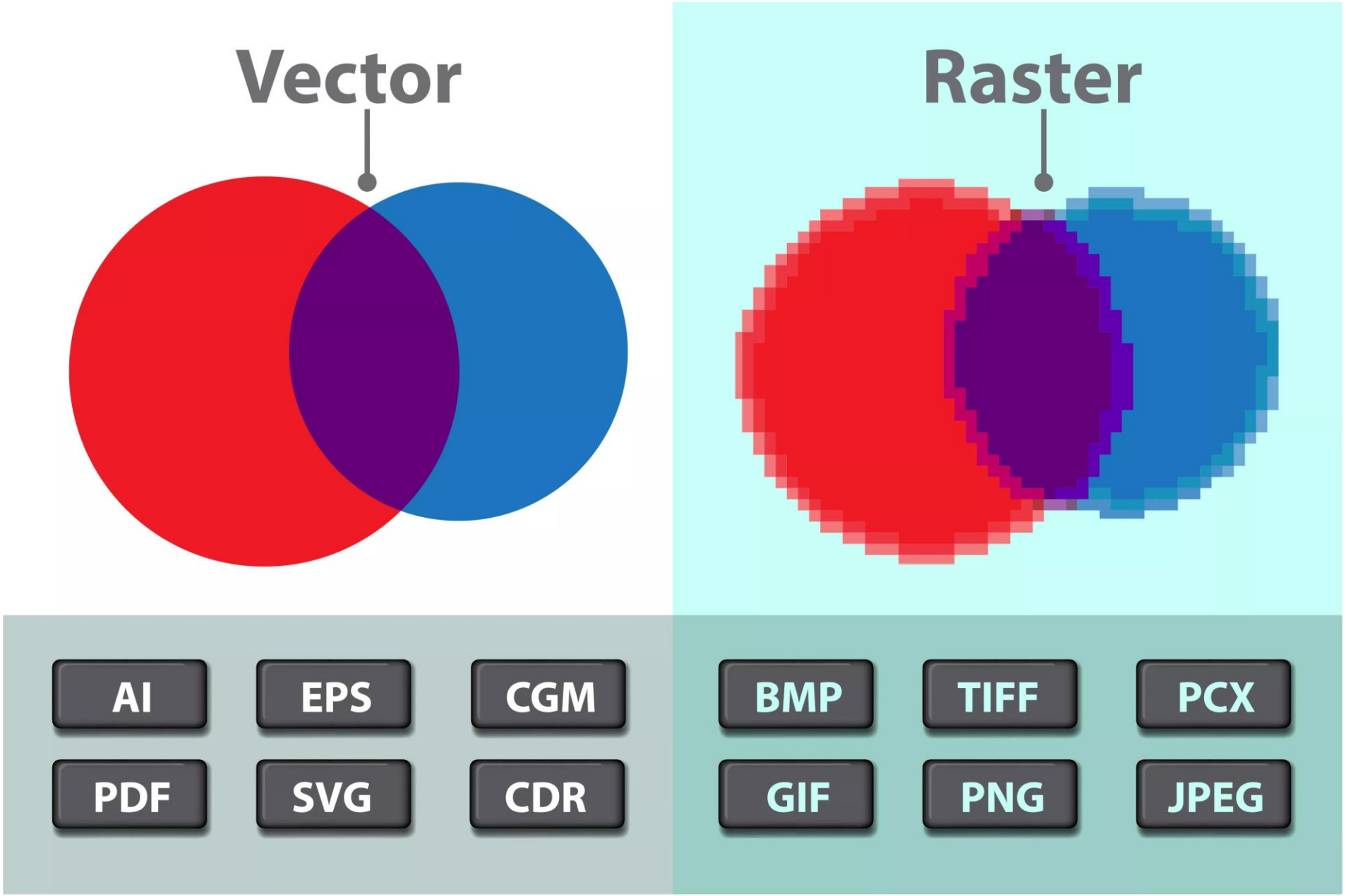
FIGURE 7-5. Browsers resize images to match the provided **width** and **height** values, but you should not resize images this way.

Vector-based Image

SECTION 6


RASTER


VECTOR


Vector
Raster
AI
EPS
CGM
PDF
SVG
CDR
BMP
TIFF
PCX
GIF
PNG
JPEG

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 300 180">
  <rect width="300" height="180" fill="purple" rx="20" ry="20"/>
  <text x="40" y="114" fill="yellow" font-family="'Verdana-Bold'"
font-size="72">
    hello!
  </text>
</svg>
```

FIGURE 7-7. A simple SVG made up of a rectangle and text.

```
<svg xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 300 180">
  <rect width="300" height="180"
        fill="purple" rx="20" ry="20"/>
  <text x="40" y="114" fill="yellow"
        font-family="'Verdana-Bold’”
        font-size="72">
  hello!
  </text>
</svg>
```

svg tag

Embedded with the img Element

SVG text files saved with the *.svg* suffix (sometimes referred to as a standalone SVG) can be treated as any other image, including placing it in the document by using the **img** element. You're an expert on the **img** element by now, so the following example should be clear:

```
<img src="/images/circle.svg"
     alt="">
```

Embedded with the img Element

Pros and cons

- The advantage to embedding an SVG with img is that it is universally supported in browsers that support SVG.
- This approach works fine when you are using a standalone SVG as a simple substitute for a GIF or a PNG

- You **cannot** apply styles to the items within the SVG by using an **external style sheet**, such as a .css file applied to the whole page. The .svg file may include its own internal style sheet using the style element, however, for styling the elements within it. You can also apply styles to the img element itself.
- You **cannot** manipulate the elements within the SVG with **JavaScript**, so you lose the option for interactivity. Scripts in your web document can't see the content of the SVG, and scripts in the SVG file do not run at all.
- Other interactive effects, like links or **:hover** styles, are never triggered inside an SVG embedded with img as well.
- You can't use any external files, such as embedded images or web fonts, within the SVG.

Embedded with the img Element

Pros and cons

Inline in the HTML Source

```
<p>This summer, try making pizza
<svg xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 72 72" width="100‟ height="100">
  <circle fill="#D4AB00" cx="36" cy="36" r="36"/>
  <circle opacity=".7" fill="#FFF" stroke="#8A291C"
          cx="36.1" cy="35.9‟ r="31.2"/>
  <circle fill="#A52C1B" cx="38.8" cy="13.5" r="4.8"/>
  <circle fill="#A52C1B" cx="22.4" cy="20.9" r="4.8"/>
  <circle fill="#A52C1B" cx="32" cy="37.2" r="4.8"/>
  <circle fill="#A52C1B" cx="16.6" cy="39.9" r="4.8"/>
  <circle fill="#A52C1B" cx="26.2" cy="53.3" r="4.8"/>
  <circle fill="#A52C1B" cx="42.5" cy="27.3" r="4.8"/>
  <circle fill="#A52C1B" cx="44.3" cy="55.2" r="4.8"/>
  <circle fill="#A52C1B" cx="54.7" cy="42.9" r="4.8"/>
  <circle fill="#A52C1B" cx="56" cy="28.3" r="4.8"/>
</svg>
on your grill.
</p>
```

Inline in the HTML Source

FIGURE 7-8. This pizza image is an SVG made up of 11 **circle** elements. Instead of an **img** element, the SVG source code is placed right in the HTML document with an **svg** element.

Embedded with the object Element

The opening **object** tag specifies the media type (an **svg+xml** image) and points to the file to be used with the **data** attribute. The **object** element comes with its own fallback mechanism—any content within the **object** gets rendered if the media specified with **data** can't be displayed. In this case, a PNG version of the image will be placed with an **img** if the *.svg* is not supported or fails to load:

```
<object type="image/svg+xml" data="pizza.svg">
    <img src="pizza.png" alt="pizza">
</object>
```

Embedded with the object Element

There is one catch, however. Some browsers download the fallback image even if they support SVG and don't need it. Useless downloads are not ideal.

The workaround is to make the fallback image a CSS background image in an empty **div** container. Unfortunately, it is not as flexible for scaling and sizing, but it does solve the extra download issue.

```
<object type="image/svg+xml"
        data="pizza.svg">
   <div style="background-image:
   url(pizza.png); width 100px;
   height: 100px;"
   role="img" aria-label="pizza">
</object>
```

Used as a Background Image with CSS

I know that this is an HTML chapter, but I'd be remiss if I didn't at least mention that SVGs can be used as background images with CSS. This style rule example puts a decorative image in the background of a **header**:

```
header {

background-image: url(/images/decorative.svg);

}
```

Use of map for Regional Links

SECTION 7

The Syntax for map

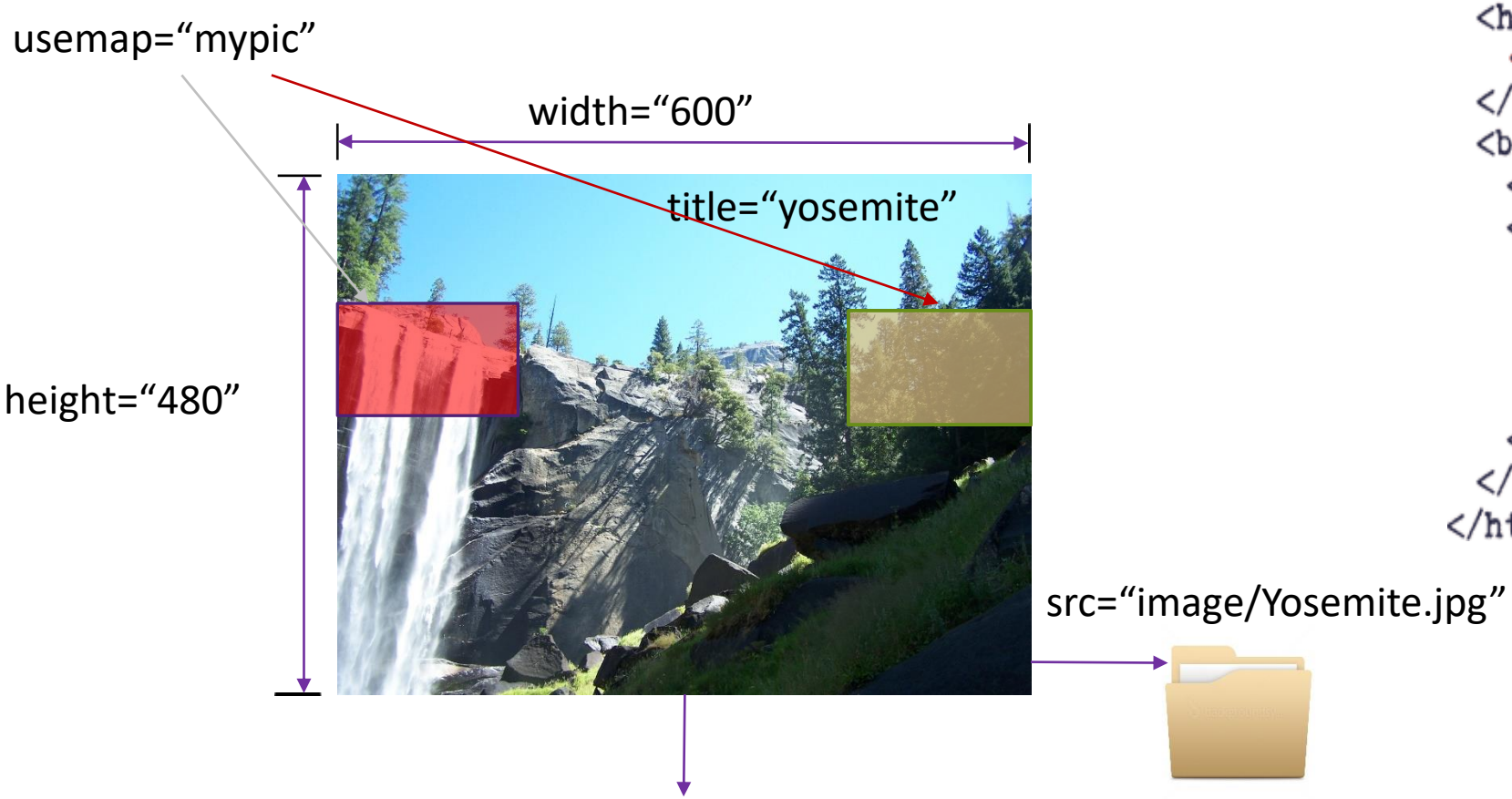
```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap"
     width="400" height="379">

<map name="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer"
        href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone"
        href="phone.htm">

  <area shape="circle" coords="337,300,44" alt="Cup of coffee"
        href="coffee.htm">
</map>
```

(x1, y1, x2, y2)

<img src="Yosemite.jpg" width="600" height="480" title="Yosemite" alt="Nevada Falls" usemap="#mypic">




```
<!DOCTYPE html>
<html>
  <head>
    .
  </head>
  <body>
    <img src="" alt="" usemap="#mypic">
    <map name="mypic">
      <area shape="rect" coords="x1,y1,x2,y2"
      href="" alt="">
      <area shape="rect" coords="x2,y1,x3,y2"
      href="" alt="">
    </map>
  </body
</html>
```

Demonstration Program

USEMAP.HTML


Insta

Notes for Image Tag

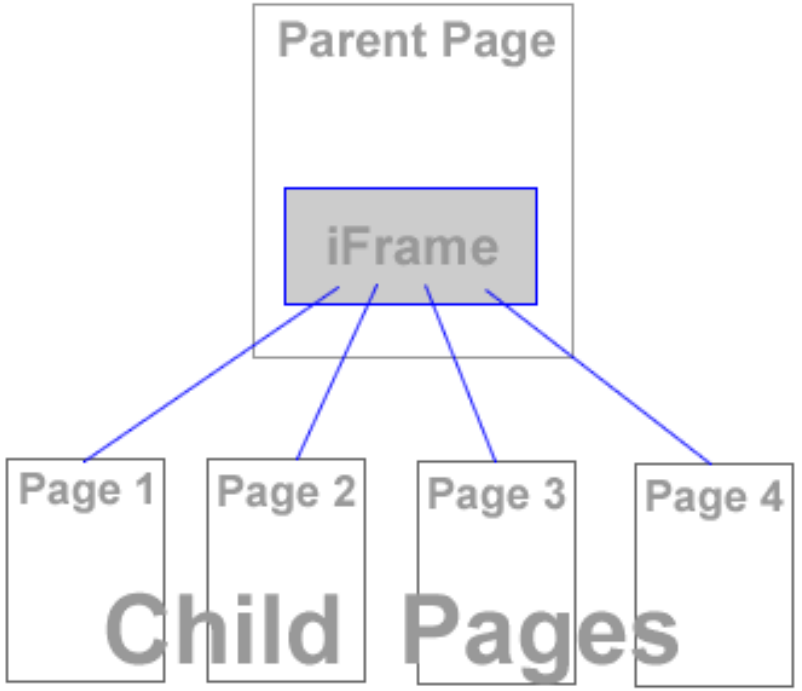
- The unit for img tag is pixels (px)
- In html5 <figure> tag and <figcaption> tag can be used to include a image file and to provide a figure caption for it.
- Do Ex. 7.1

Page in iframe

SECTION 8

Purpose of Text iframe

- Listing of Computer Code
- Listing of an external data file
- Another page view



iframe tag

<iframe src="html file" width="number" height="number">

Responsive Image Markup

SECTION 9

Responsive Image Markup

The responsive image attributes and elements address the following four basic scenarios:

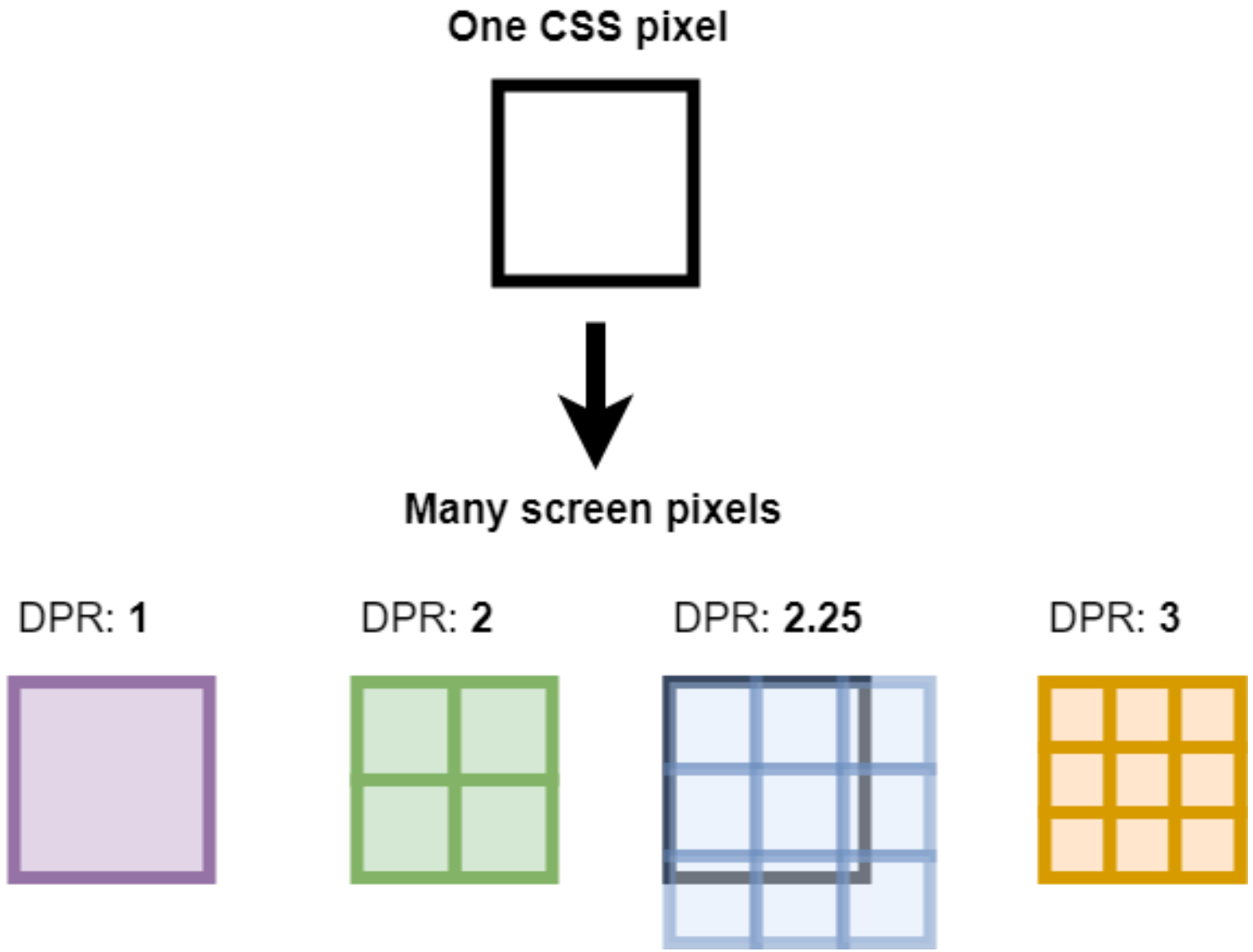
- Providing extra-large images that look crisp on high-resolution screens
- Providing a set of images of various dimensions for use on different screen sizes
- Providing versions of the image with varying amount of detail based on the device size and orientation (known as the art direction use case)
- Providing alternative image formats that store the same image at much smaller file sizes

High-Density Displays (x-descriptor)

Devices use a measurement called a reference pixel for layout purposes.

It should come as no surprise that it’s not so straightforward today. Manufacturers have been pushing screen resolutions higher and higher in an effort to improve image quality. The result is that device pixels have been getting smaller and smaller, so small that our images and text would be illegibly tiny if they were mapped one-to-one.

Device Pixel Ratio



Device Pixel Ratio

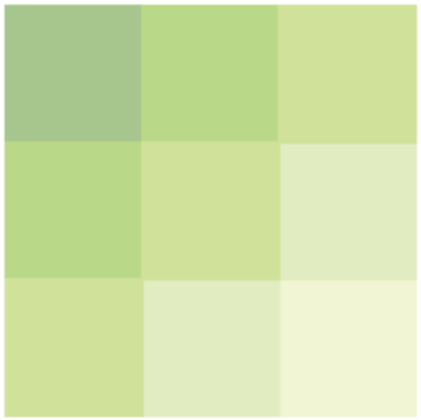
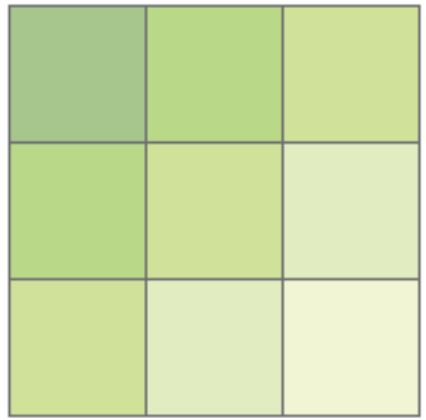
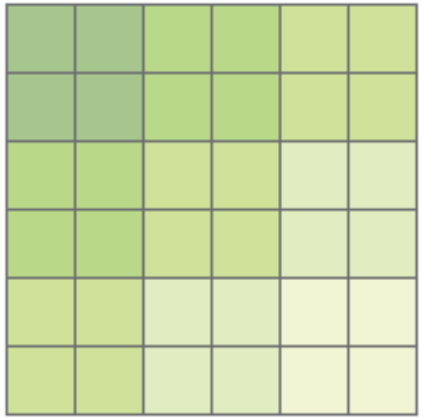


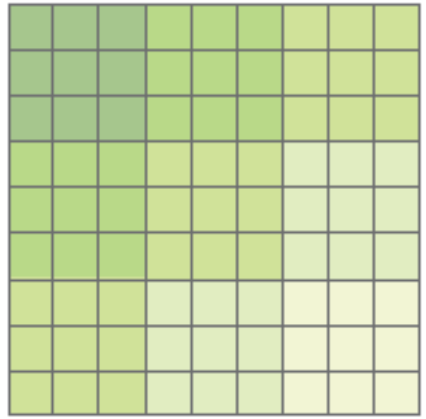
Image or object =
3 x 3 *reference or CSS pixels*



1:1 device-pixel-ratio (1x)
3 x 3 *device pixels*, indicated by grid



2:1 device-pixel-ratio (2x)
6 x 6 device pixels



3:1 device-pixel-ratio (3x)
9 x 9 device pixels

FIGURE 7-10. Device pixels compared to CSS/reference pixels.

This sample shows the structure of a **srcset** value:

```
    srcset="image-URL #x, image-URL #x"

<img src="image-URL" alt="" srcset="image-
URL #x, image-URL #x">

<img src="/images/turkey-200px.jpg" alt=""
     srcset="/images/turkey-400px.jpg 2x,
             /images/turkey-600px.jpg 3x" >

<img src="/images/turkey-200px.jpg" alt=""
     srcset="/images/turkey-400px.jpg 2x,
             /images/turkey-600px.jpg 3x" >
```

Introducing srcset

Variable-Width Images (w-descriptor)

- When you're designing a responsive web page, chances are you'll want image sizes to change based on the size of the browser viewport (see Note). This is known as a viewport-based selection.
- And because you are the type of web developer who cares about how fast pages display, you'll want to limit unnecessary data downloads by providing appropriately sized images.

Variable-Width Images (w-descriptor)

- To achieve this goal, use the **srcset** and **sizes** attributes with the **img** element.
- As we saw in previous examples, the **srcset** gives the browser a set of image file options, but this time, it uses a w-descriptor (width descriptor) that provides the *actual pixel width* of each image. Using **srcset** with a w-descriptor is appropriate when the images are identical except for their dimensions (in other words, they differ only in scale). Here's an example of a **srcset** attribute that provides four image options and specifies their respective pixel widths via w-descriptors. Note again that the whole list is in a single set of quotation marks:

Variable-Width Images (w-descriptor)

```
srcset="strawberries-480.jpg 480w,
        strawberries-960.jpg 960w,
        strawberries-1280.jpg 1280w,
        strawberries-2400.jpg 2400w"
```

Using the sizes attribute

```
<img src="strawberries-640.jpg"
alt="baskets of ripe strawberries"
srcset="strawberries-480.jpg 480w,
strawberries-960.jpg 960w,
strawberries-1280.jpg 1280w,
strawberries-2400.jpg 2400w"
sizes="100vw">
```

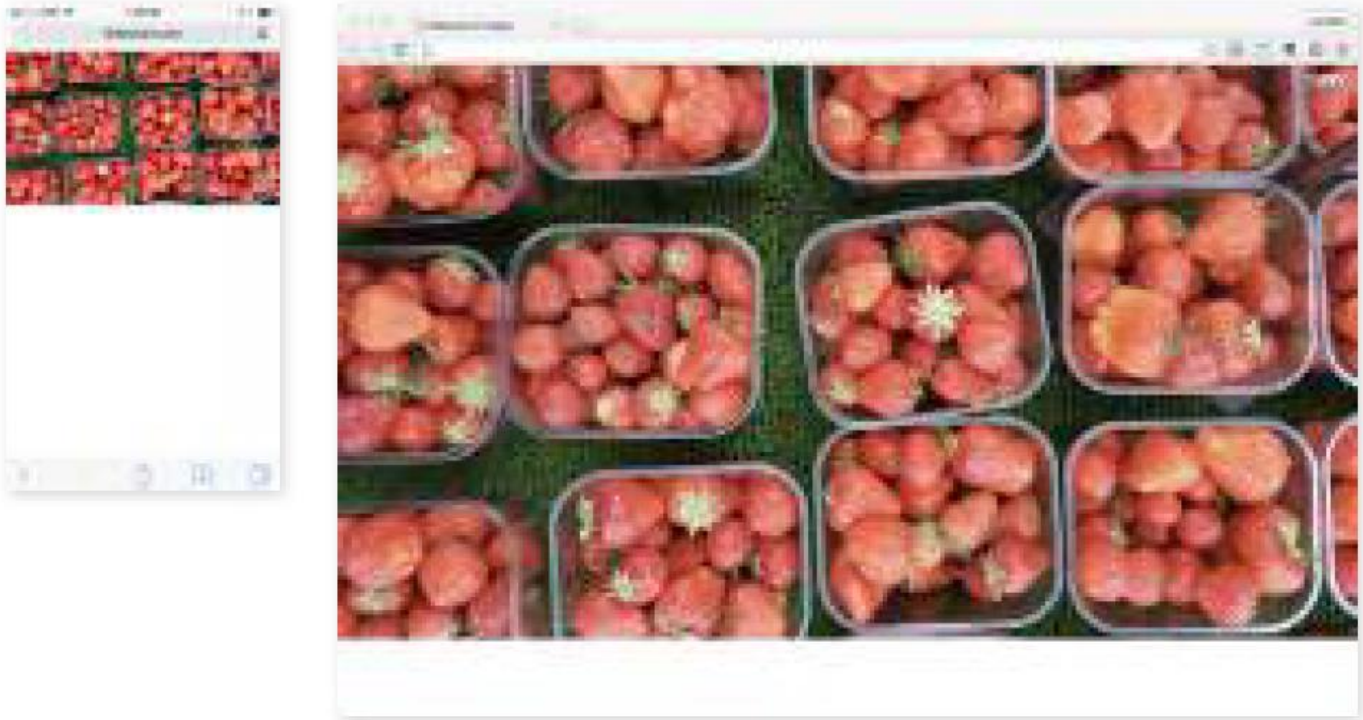


FIGURE 7-11. The image fills 100% of the viewport width, regardless of its size.