Faculty of Science
Department of Computing and Information Systems (COIS)

COIS 2830

Multimedia and Design

Danny Papagiannis

Andy Reid

Thomas Hughes

Intro To HTML

As in, how to make your webpage

Agenda

1. Elements, Attributes, Troubleshooting
2. Adding Links
3. Block Elements
4. Adding Images

What you should use

- https://developer.mozilla.org/en-US/docs/Learn/HTML
- That's from the guys who make firefox and it's probably the best reference in terms of teaching you good habits etc.

- https://www.w3schools.com/html/
- Is probably good enough for COIS 2830H, but it teaches some bad habits, especially with forms (which we don't use in 2830).
- (Forms are how you fill in data on a website)

- https://v.redd.it/eot1zxfqcrj31

1

Elements, Attributes, Troubleshooting

OVERVIEW

- How markup works: Elements and attributes
- Minimal HTML document structure
- Identifying text elements: Block and inline
- Troubleshooting and validating HTML

Content Without Markup

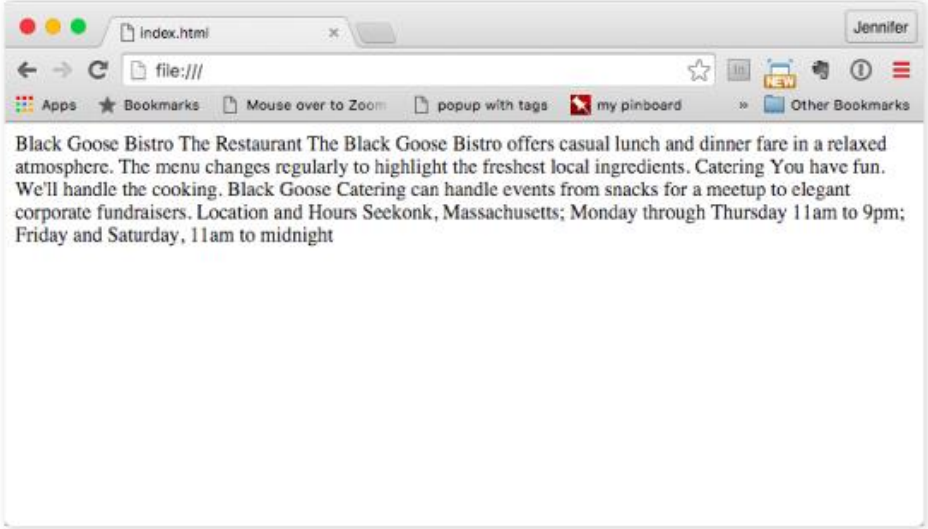
Without HTML markup to describe content structure, text runs together; line breaks are ignored:

```
Black Goose Bistro


The Restaurant
The Black Goose Bistro offers casual lunch
and dinner fare in a relaxed atmosphere. The
menu changes regularly to highlight the
freshest local ingredients.


Catering
You have fun. We'll handle the cooking. Black
Goose Catering can handle events from snacks
for a meetup to elegant corporate
fundraisers.


Location and Hours
Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm; Friday
and Saturday, 11am to midnight
```



What Browsers Ignore

- Multiple character spaces (white space)
- Line breaks (carriage returns)
- Tabs
- Unrecognized markup

- (Sri note: Sometimes)

Markup Basics

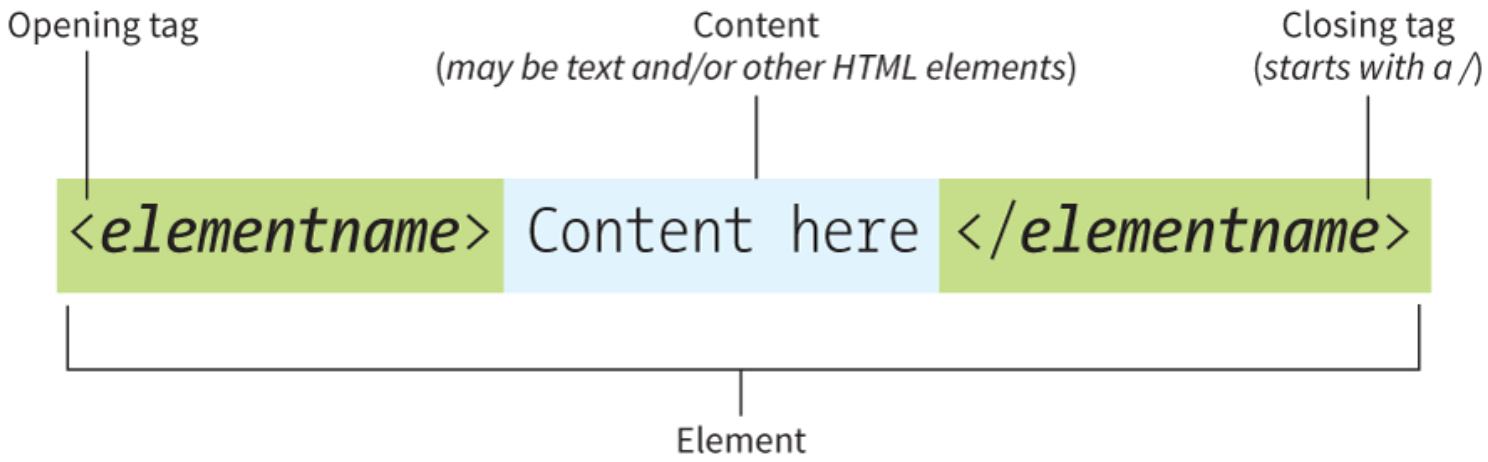
Text must be marked up meaningfully and accurately (**semantically**) with HTML tags:

- Browsers need markup to display content correctly.
- Markup makes content elements available to scripts and style rules.
- Markup aids search engines.

Anatomy of an HTML Element

tag
The element name in angle brackets

element
The content and its markup (start and end tags)



Example: `<h1>Black Goose Bistro</h1>`

Some Elements Are Empty

Some elements have no content and provide a simple directive. They are called **empty elements**:

```
<element-name>
```

Example: The **br** element inserts a line break.

```
<p>1005 Gravenstein Highway North<br>Sebastopol, CA 95472</p>
```

Attributes

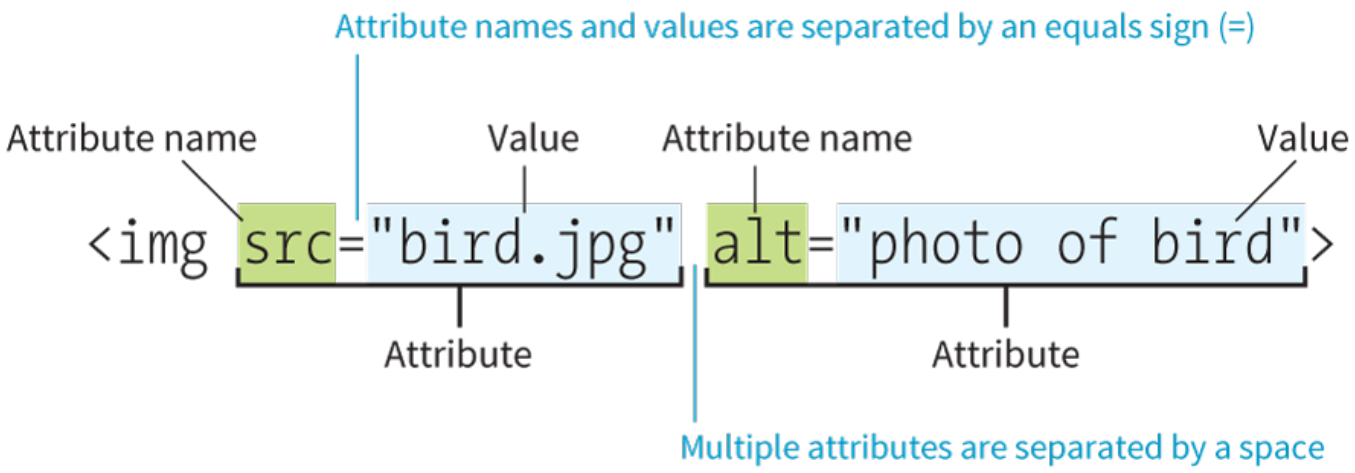
Attributes are instructions that clarify or modify an element. They appear in the opening tag after the element name:

```
<element attribute="value">Content</element>
```

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

Attributes (cont'd)

There can be more than one attribute in a tag:



They are separated by spaces and can go in any order.

Attributes (cont'd)

- Most attributes take values, which follow an = sign; some are single descriptive words.
- A value might be a number, word, string of text, URL, or measurement.
- Quotation marks aren't strictly required but are recommended for consistency.
- Single or double quotation marks are okay.
- Attribute names and values are defined in the HTML specification.
- Some attributes are required.

Nesting Elements

Putting elements inside other elements is called **nesting**. Make sure closing tags don't overlap:

```
<div>
  <h1>Headline</h1>
  <p>This is <em>emphasized</em> text.</p>
</div>
```

HTML Comments

To leave a note in an HTML document, mark it up as a **comment**:

```
<!-- start global navigation -->
<ul>
…
</ul>
<!-- end global navigation
and begin the main header
for content page -->
```

Text between `<!--` and `-->` won't display in the browser. Keep in mind that they are still visible in the source.

Minimal Document Structure

It is recommended that HTML documents have the following minimal structure:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Title here</title>
  </head>
  <body>
    <!-- Page content goes here.-->
  </body>
</html>
```

Minimal Document Structure (cont'd)

The **DOCTYPE declaration** lets browsers know that the document is written according to the HTML5 specification:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Title here</title>
  </head>
  <body>
    [Page content goes here.]
  </body>
</html>
```

Minimal Document Structure (cont'd)

The **`html`** element is the **root element** that contains all the elements in the document:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Title here</title>
  </head>
  <body>
    [Page content goes here.]
  </body>
</html>
```

Minimal Document Structure (cont'd)

The **`head`** element contains elements that pertain to the document and are not rendered as content, such as **title**, **metadata**, **style sheets**, and **scripts**:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Title here</title>
  </head>
  <body>
    [Page content goes here.]
  </body>
</html>
```

Minimal Document Structure (cont'd)

`meta` elements provide document **metadata**. In this case, it says that the document uses an expanded **character set** (UTF-8). It could also provide keywords, author information, publishing status, and more:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Title here</title>
  </head>
  <body>
    [Page content goes here.]
  </body>
</html>
```

Minimal Document Structure (cont'd)

The **`title`** element is required. Titles display in the browser tab and bookmark lists, and are used by search engines:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Title here</title>
  </head>
  <body>
    [Page content goes here.]
  </body>
</html>
```

Minimal Document Structure (cont'd.)

The **body** element contains the content of the document. The content is everything you want to show up in the browser window:

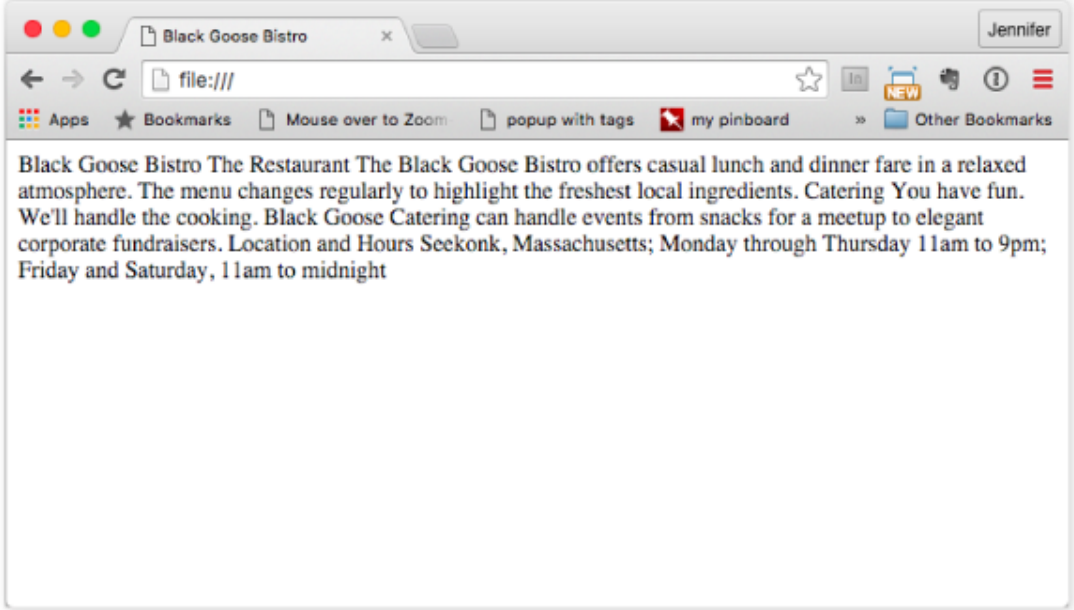
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Title here</title>
  </head>
  <body>
    [Page content goes here.]
  </body>
</html>
```

A Structured Document

With the document structure in place, there is a title in the browser tab, but the content is still unstructured.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Black Goose Bistro</title>
</head>
<body>
Black Goose Bistro

The Restaurant
The Black Goose Bistro offers casual
lunch and dinner fare in a relaxed
atmosphere. The menu changes regularly
to highlight the freshest local
ingredients.
…
Location and Hours
Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm;
Friday and Saturday, 11am to midnight
</body>
</html>
```



Marking Up Content

- The purpose of HTML is to add meaning and structure to the content. This is called **semantic markup**.
- It is **not** intended to describe how the content should look (its presentation).
- The way elements relate to one another forms an outline-like structure called the **DOM (Document Object Model)**.
- The DOM is the foundation on which you apply styles and scripts.

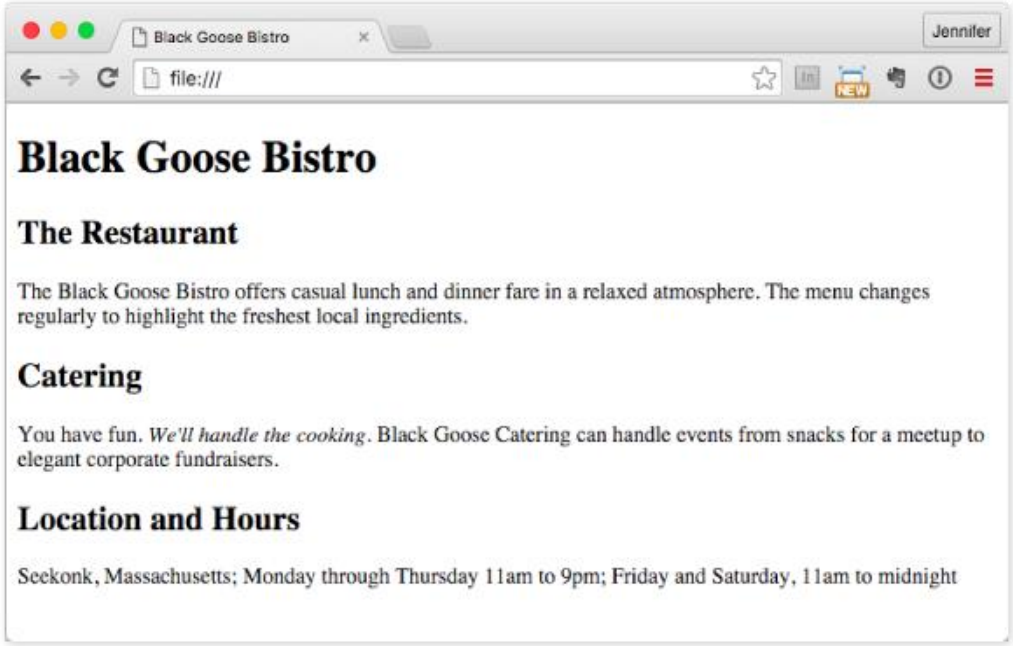
Basic Text Elements

With headings (<h1>) and paragraphs (<p>) identified, the browser can display the content appropriately:

```
…
<body>
<h1>Black Goose Bistro</h1>

<h2>The Restaurant</h2>
<p>The Black Goose Bistro
offers casual lunch and
dinner fare in a relaxed
atmosphere. The menu changes
regularly to highlight the
freshest local
ingredients.</p>

<!--more content-->
</body>
```



Block and Inline Elements

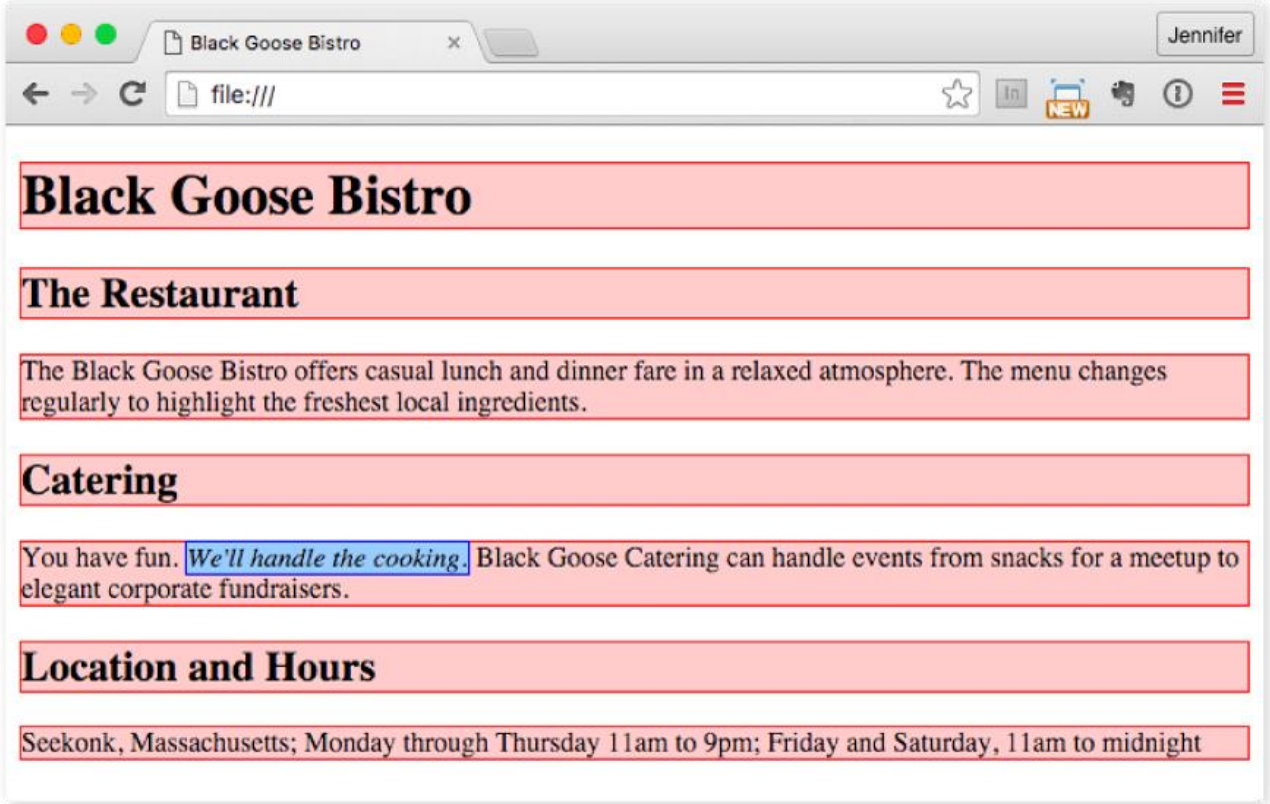
Block elements
Major components of the page that display like rectangular blocks stacking up on the page

Example: headings, paragraphs, long quotations

Inline elements
Appear within the text flow of a block element

Example: emphasized text, links, abbreviations

Block and Inline Elements (cont'd)



Block elements (headings, paragraphs) outlined in RED.

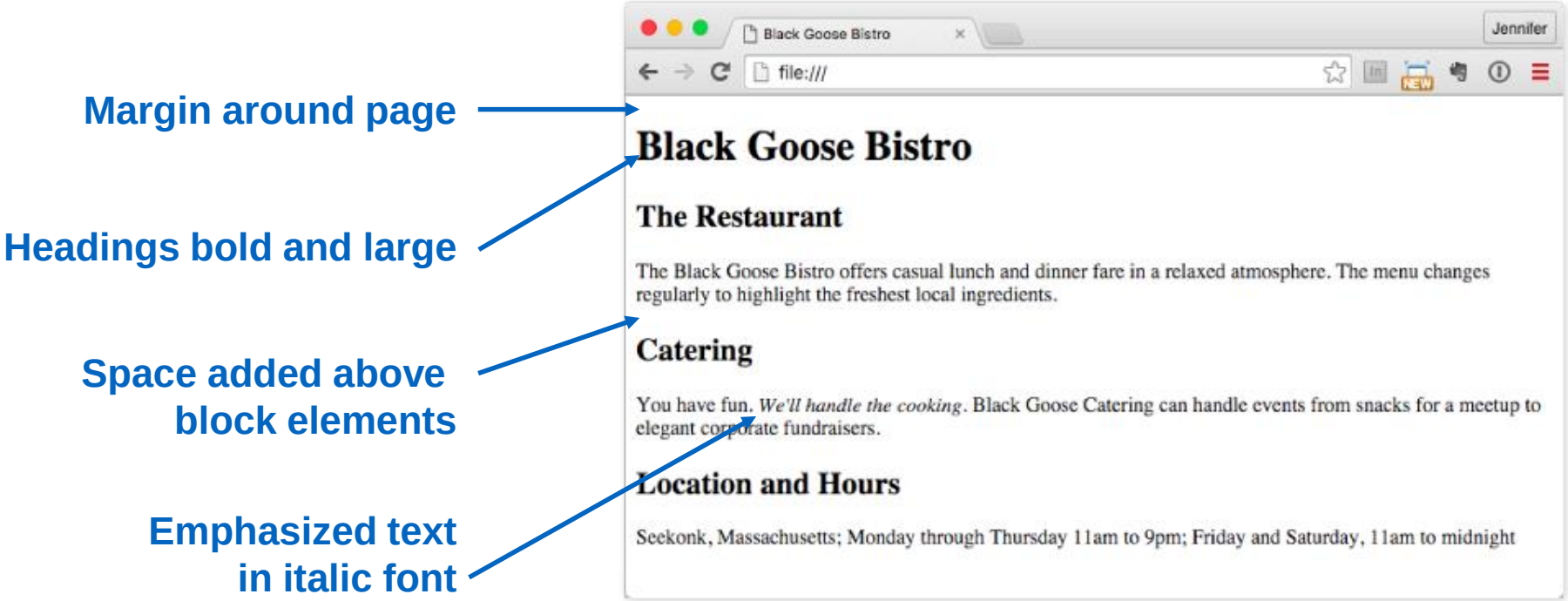
Inline element (emphasized text) outlined in BLUE.

Style Sheets

- HTML only describes structure, not presentation
- Presentation is controlled by style sheets (CSS)
- Browsers have their own style sheets (user agent style sheets) with default styles for HTML elements
- You can write your own styles to override the default styles

Style Sheets (cont'd)

The browser parsed the markup and used its built-in style sheet to format the text elements in the example:



Troubleshooting HTML

Small mistakes and missing characters can cause HTML documents to “break.”

Common mishaps:

- Missing closing tag (or / in closing tag)
- Missing closing bracket in a tag
- Missing quotation mark around an attribute value
- Not saving your document before viewing changes in the browser

Troubleshooting (cont'd)

When a slash is omitted, the browser doesn't know when the element ends:

```
<h2>Catering</h2>
<p>You have fun. <em>We'll handle the cooking.<em> Black Goose
Catering can handle events from snacks for a meetup to elegant
corporate fundraisers.</p>
```

g.

Catering

You have fun. *We'll handle the cooking. Black Goose Catering can handle events from snacks for a meetup to elegant corporate fundraisers.*

Location and Hours

Seekonk, Massachusetts;
Monday through Thursday 11am to 9pm;
Friday and Saturday, 11am to midnight

Troubleshooting (cont'd)

A missing end bracket makes the browser interpret all the following characters as part of the tag:

```
<h2The Restaurant</h2>
<p>The Black Goose Bistro offers casual lunch and dinner fare
in a relaxed atmosphere. The menu changes regularly to highlight
the freshest local ingredients.</p>
```

<h2The

Missing subhead

Without the bracket, all the following characters are interpreted as part of the tag, and "The Restaurant" disappears from the page.

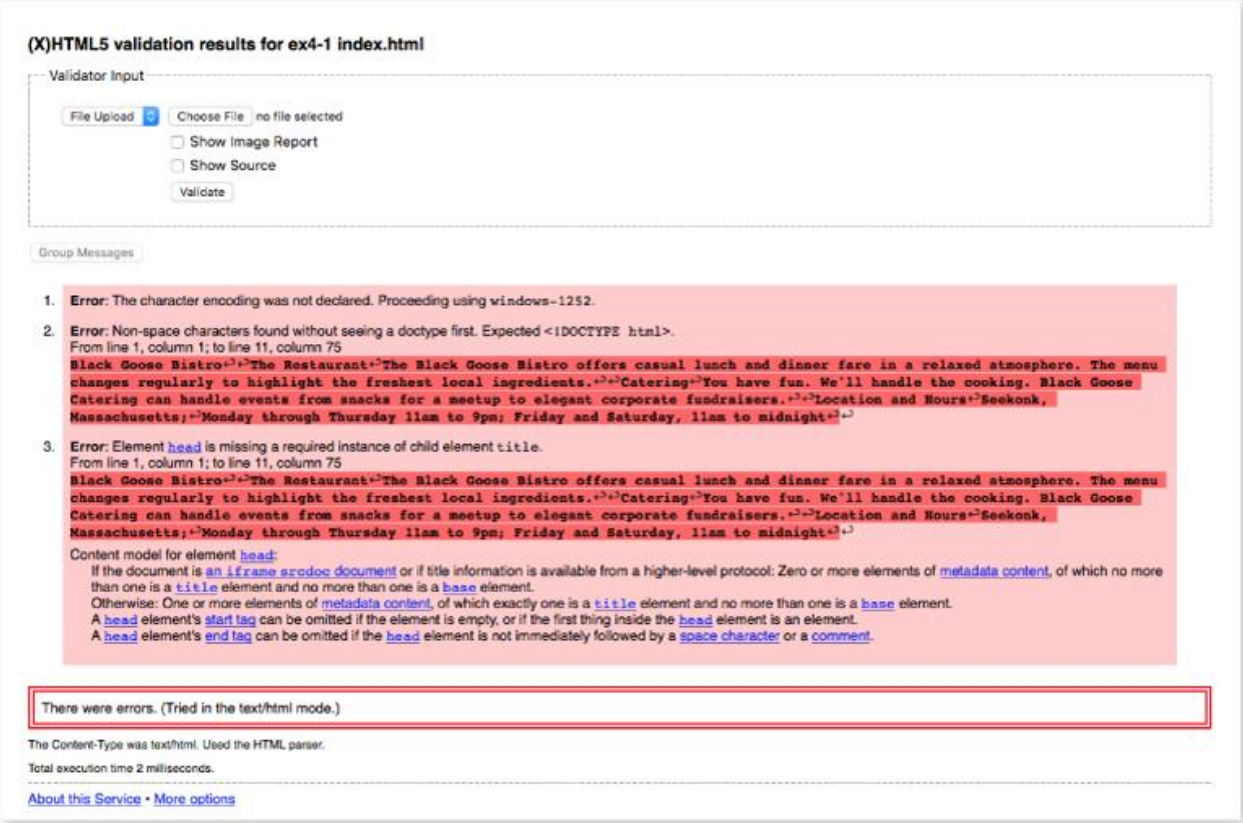
Validating Your Documents

Validate a document to make sure that you have abided by the HTML rules and that there are no errors:

- Include the DOCTYPE declaration
- Indicate the character encoding
- Include required attributes
- Don’t use non-standard elements
- Don’t mismatch tags
- Nest elements correctly (no overlaps)
- Check for typos and other minor errors

Validating Your Documents

- Use a validator tool like the one at **html5.validator.nu**.
- Upload a document or provide a link, and the validator returns an error report.



(X)HTML5 validation results for ex4-1 index.html

Validator Input

File Upload | Choose File | no file selected

Show Image Report

Show Source

Validate

Group Messages

1. **Error:** The character encoding was not declared. Proceeding using `windows-1252`.
2. **Error:** Non-space characters found without seeing a doctype first. Expected `<!DOCTYPE html>`.
   From line 1, column 1; to line 11, column 75
   `Black Goose Bistro↵↵The Restaurant↵The Black Goose Bistro offers casual lunch and dinner fare in a relaxed atmosphere. The menu changes regularly to highlight the freshest local ingredients.↵↵Catering↵You have fun. We'll handle the cooking. Black Goose Catering can handle events from snacks for a meetup to elegant corporate fundraisers.↵↵Location and Hours↵Seekonk, Massachusetts;↵Monday through Thursday 11am to 9pm; Friday and Saturday, 11am to midnight↵↵`
3. **Error:** Element `head` is missing a required instance of child element `title`.
   From line 1, column 1; to line 11, column 75
   `Black Goose Bistro↵↵The Restaurant↵The Black Goose Bistro offers casual lunch and dinner fare in a relaxed atmosphere. The menu changes regularly to highlight the freshest local ingredients.↵↵Catering↵You have fun. We'll handle the cooking. Black Goose Catering can handle events from snacks for a meetup to elegant corporate fundraisers.↵↵Location and Hours↵Seekonk, Massachusetts;↵Monday through Thursday 11am to 9pm; Friday and Saturday, 11am to midnight↵↵`
   Content model for element `head`:
   If the document is an `iframe srcdoc` document or if title information is available from a higher-level protocol: Zero or more elements of metadata content, of which no more than one is a `title` element and no more than one is a `base` element.
   Otherwise: One or more elements of metadata content, of which exactly one is a `title` element and no more than one is a `base` element.
   A `head` element's start tag can be omitted if the element is empty, or if the first thing inside the `head` element is an element.
   A `head` element's end tag can be omitted if the `head` element is not immediately followed by a space character or a comment.

There were errors. (Tried in the text/html mode.)

The Content-Type was text/html. Used the HTML parser.

Total execution time 2 milliseconds.

About this Service • More options

2

Adding Links

OVERVIEW

- The `a` element
- External links with absolute URLs
- Links with relative pathnames
- Linking within a page (fragments)
- Targeting browser windows
- Mail links

Adding Links

<a>

```
<a href="URL">Link text or image</a>
```

The **`href`** attribute provides the location (URL) of the resource.

You can link to any resource:

- HTML documents
- Images
- Movies
- PDFs
- More!

href Attributes

- **Absolute URLs** begin with the protocol (ex: `http://`).
- **Relative URLs** provide the path to a file on the same server as the document containing the link (ex: `/directory/document.html`).
- Long URLs can make the markup look complicated, but the structure is the same:

Opening anchor tag

```
<a href="https://www.amazon.com/Bequet-Gourmet-Caramel-24oz
-Celtic/dp/B00GZEU10Y/ref=sr_1_1_a_it?ie=UTF8&qid=1467055107&s
r=8-1&keywords=bequet">Bequet Caramels</a>
```

URL | Linked text | Closing anchor tag

External Links

- **External links** go to pages that are **not** on your server.
- An absolute URL is required, including "**http://**"

```
<li><a
href="http://www.foodnetwork.com">The
Food Network</a></li>
```

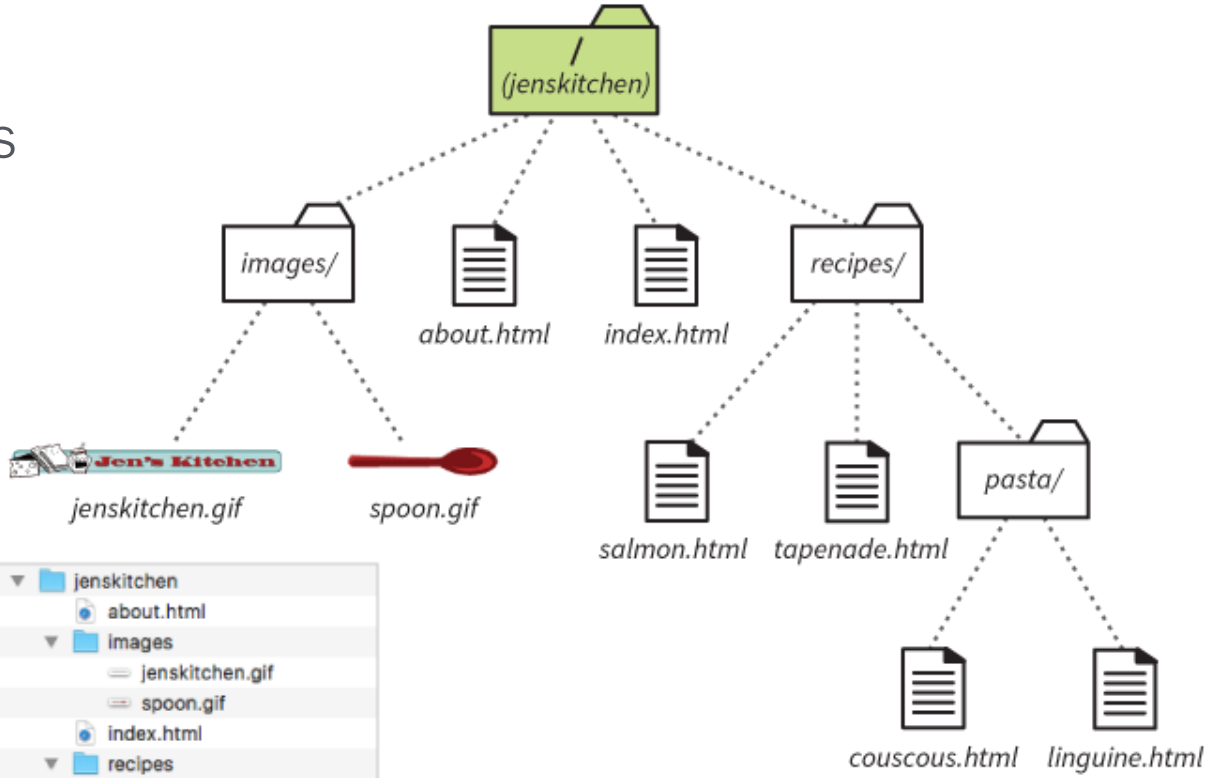
Linking on Your Own Site

- When no protocol is provided, the browser looks on the current server for the resource.
- A **relative pathname** describes how to get to the resource starting from the current document.
- Pathnames follow UNIX syntax conventions.

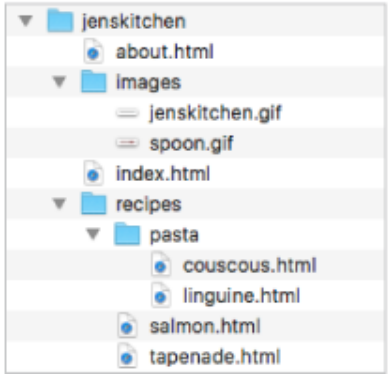
Example Server Directory Structure

The following relative pathname discussions are based on this site structure.

The root directory is called *jenskitchen*.



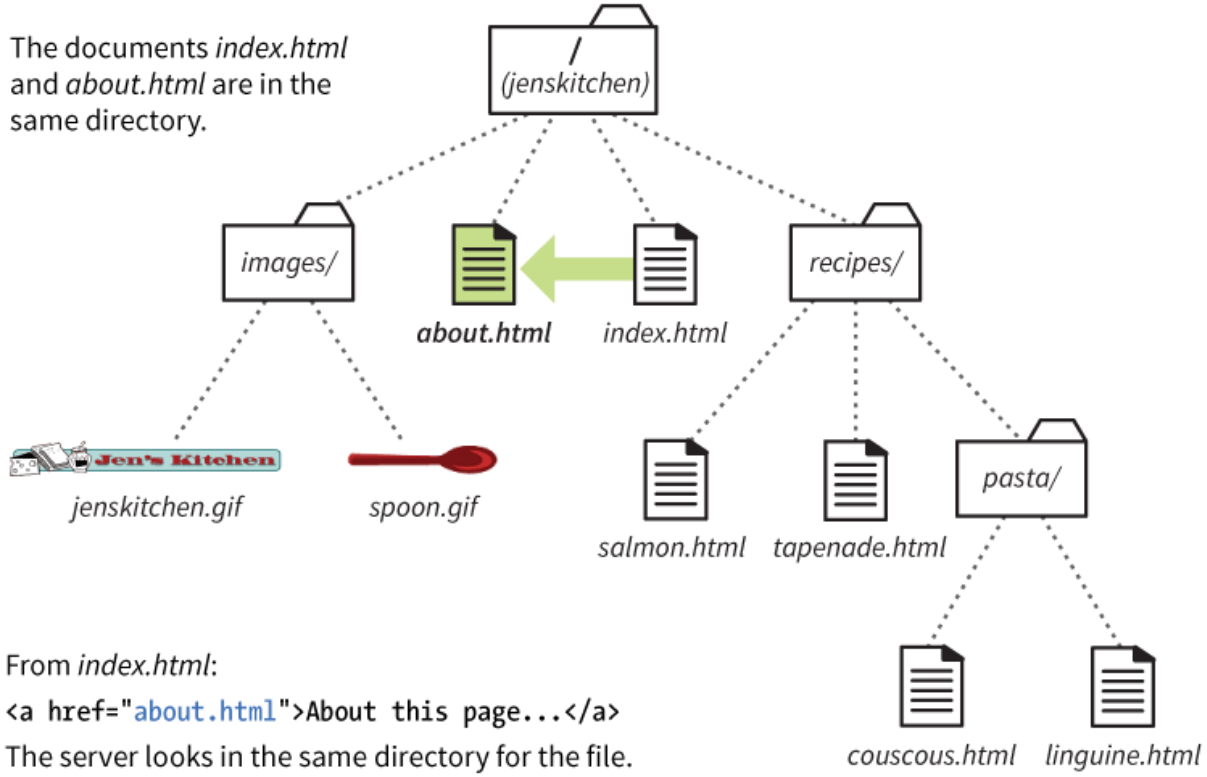
How it looks in the MacOS Finder



Linking in the Same Directory

When the linked document is in the same directory as the current document, just provide its filename:

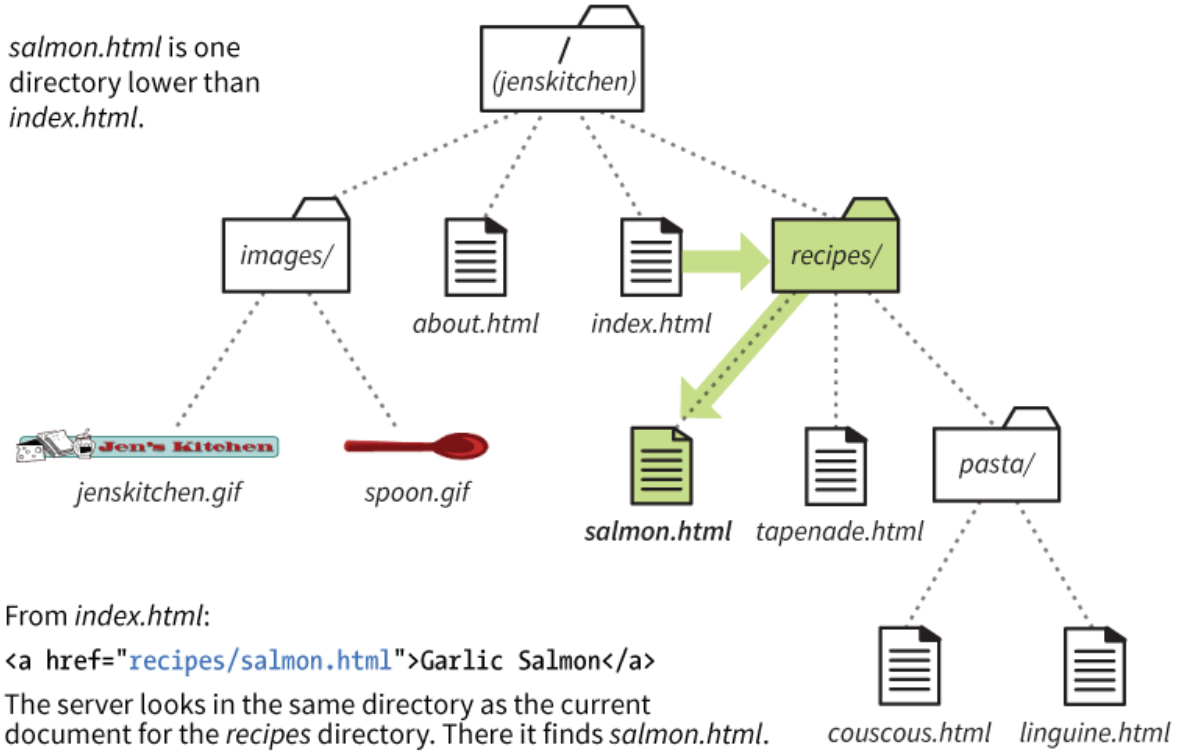
```
href="about.html"
```



Linking into a Lower Directory

If the linked file is in a directory, include the directory name in the path.

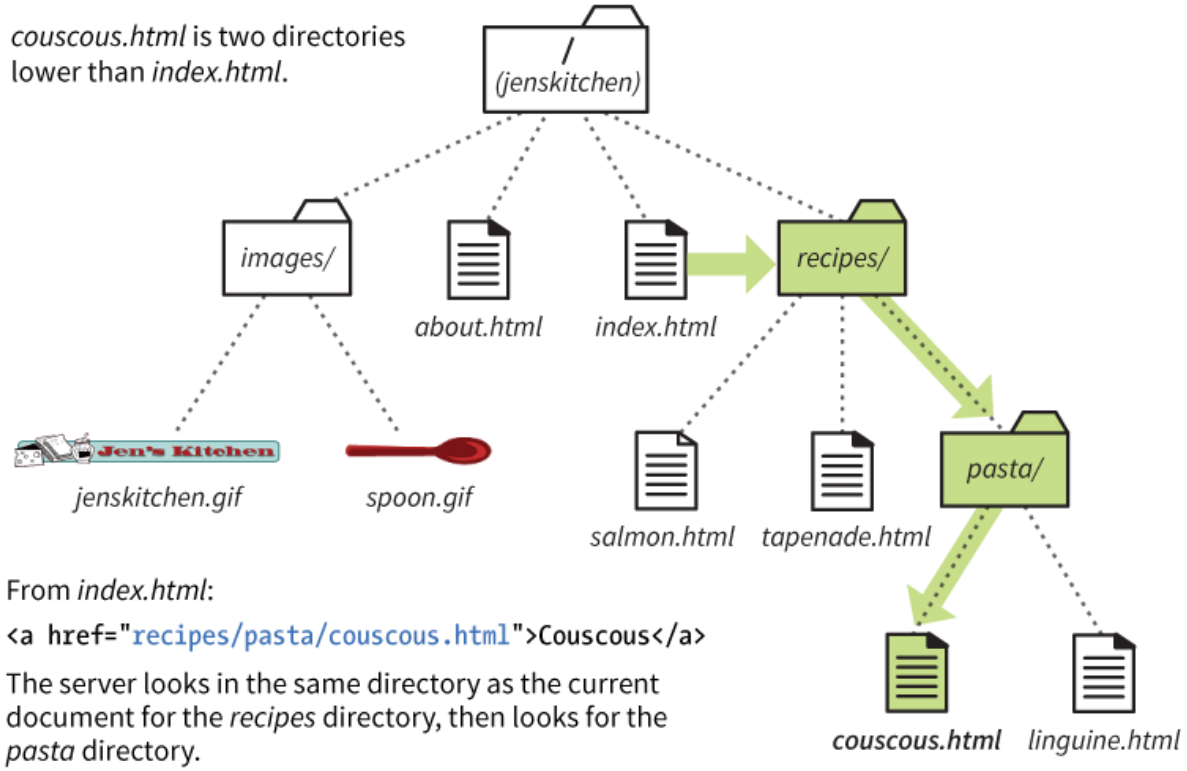
```
href="recipes/salmon.html"
```



Linking into Two Directories

Include each subdirectory name in the path to the linked document:

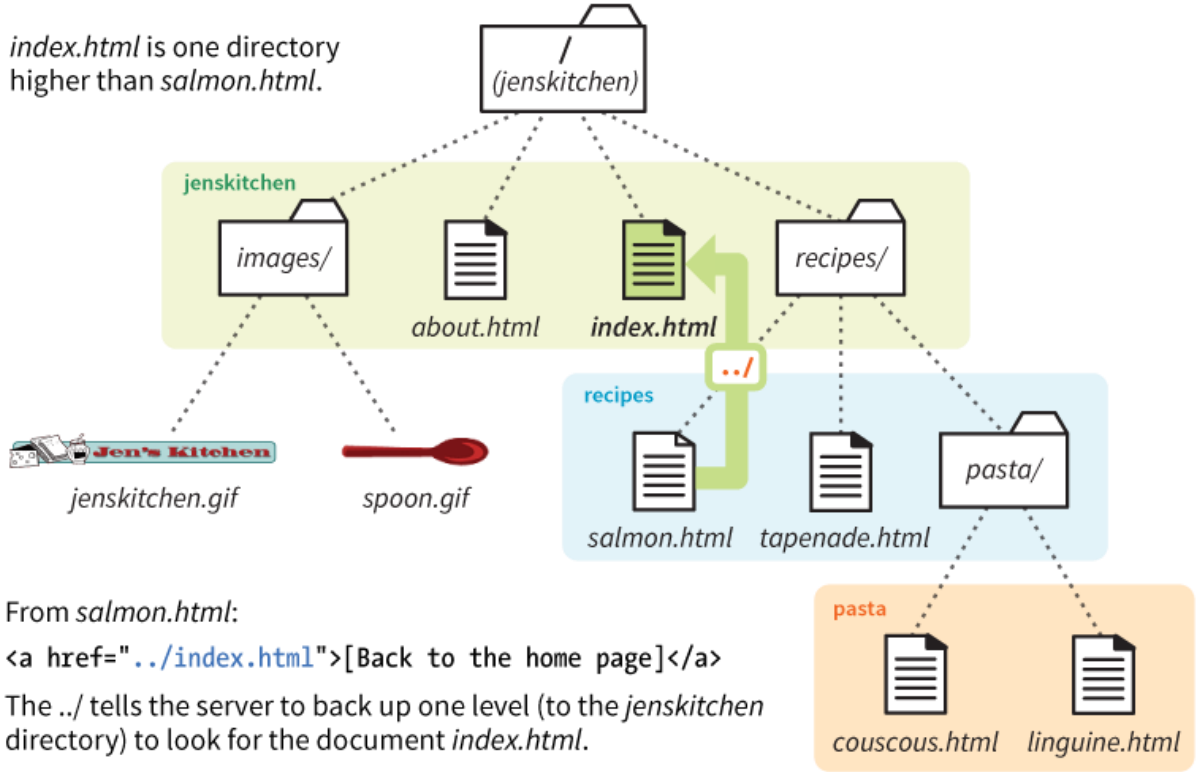
```
href="recipes/pasta/couscous.html"
```



Linking to a Higher Directory

To back up a level, the `../` stands in for the name of the higher directory:

```
href="../index.html"
```



From *salmon.html*:

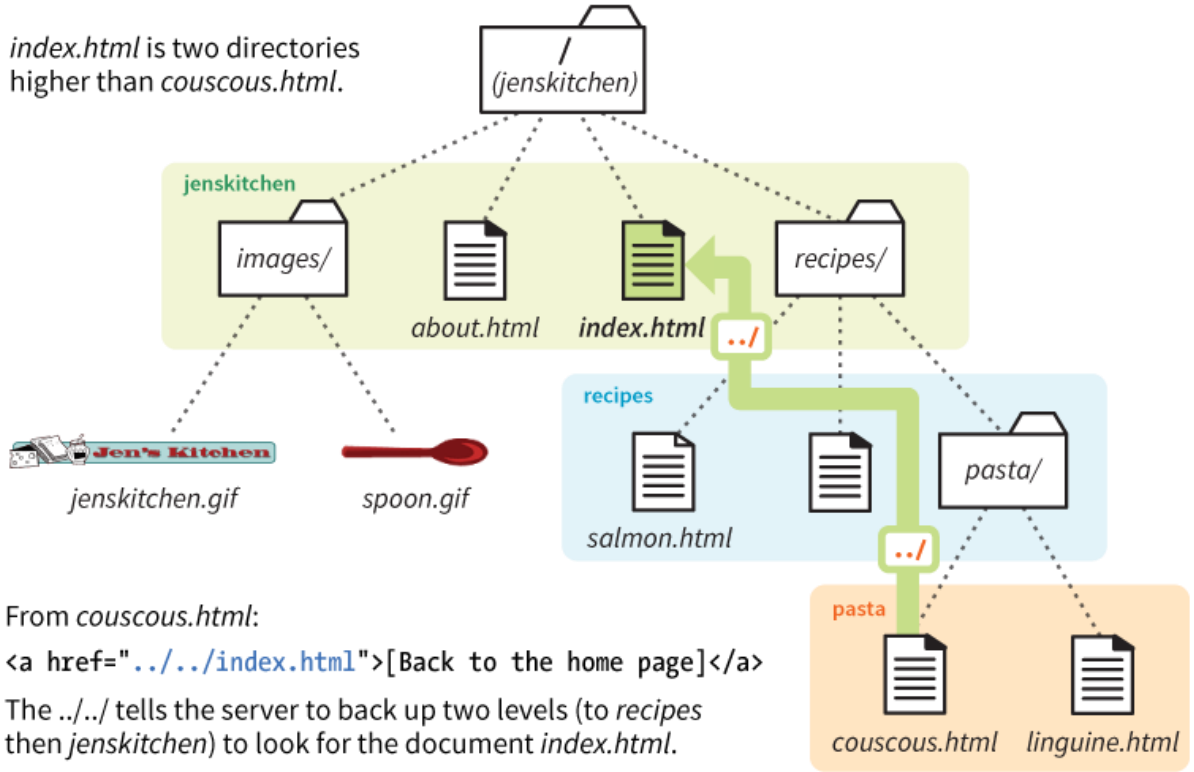
```
<a href="../index.html">[Back to the home page]</a>
```

The ../ tells the server to back up one level (to the *jenskitchen* directory) to look for the document *index.html*.

Linking Up Two Directory Levels

Include a `../` for each level you need to back up to:

```
href="../../index.html"
```



Site Root Relative Paths

Starting the path with a slash / means the pathname starts at the root directory. You don't need to write the name of the directory.

```
href="/recipes/salmon.html"
```

NOTE: Site root relative URLs are more flexible because they work from any document on the site.

DOWNSIDE: They won't work on your own computer because the / will be relative to your hard drive. You'll need to test it on the actual web server.

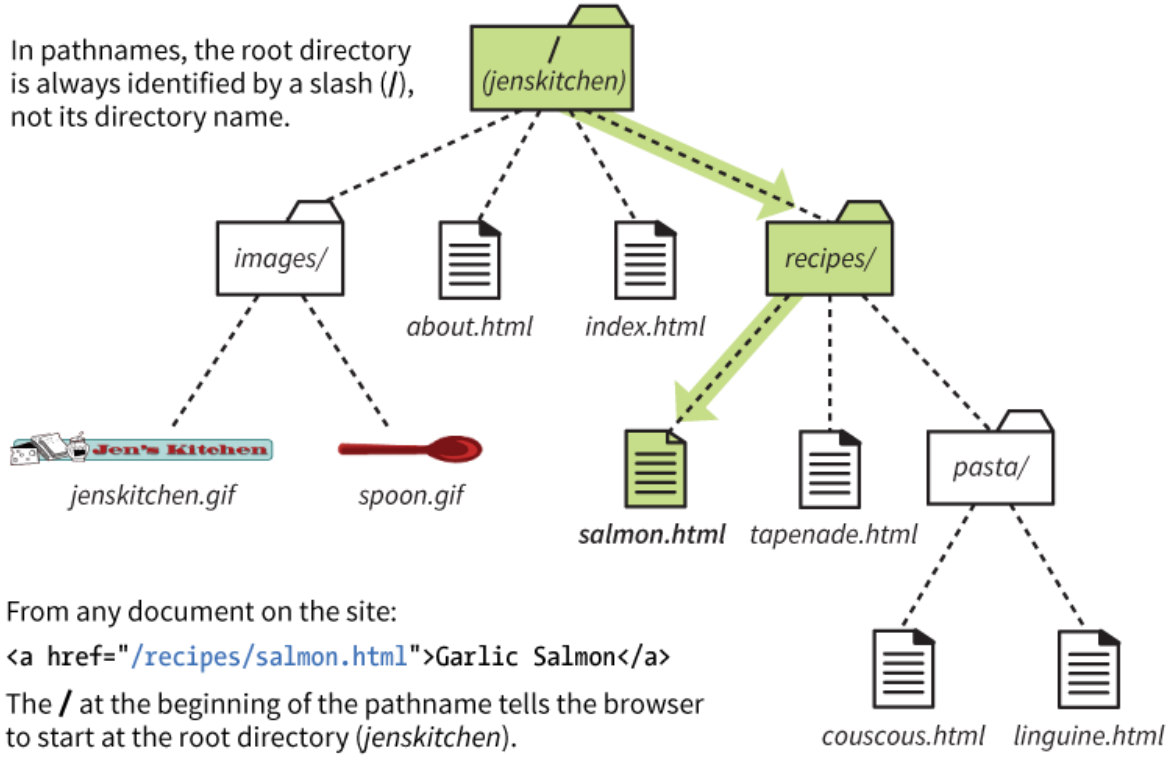


Image src Pathnames

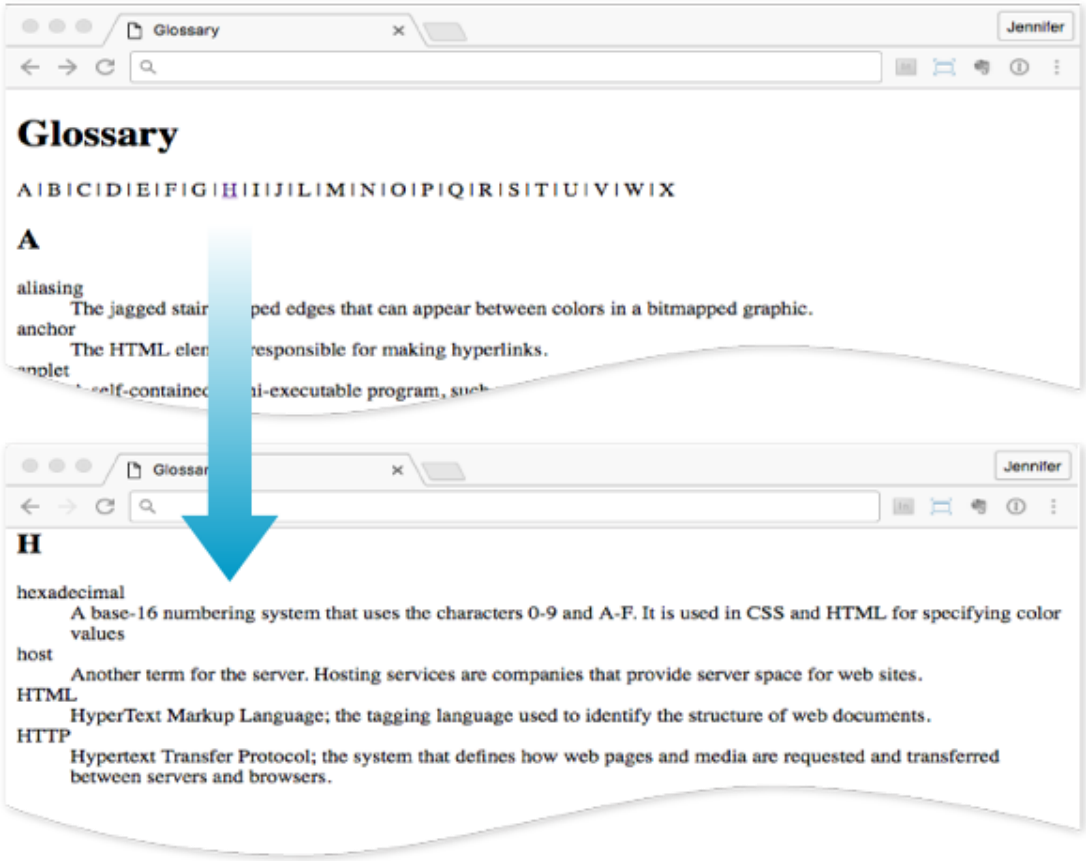
Relative pathnames are also commonly used to point to image files with the **`src`** attribute:

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

Linking Within a Page (Fragments)

Linking to a specific point on a web page is called linking to a document fragment.

This is useful for providing links down to content from the top of a long document.



Linking to a Fragment

For example, to create a link from the letter H in a list at the top of the page to the “H” heading farther down in the document:

Step 1: Identify the target destination.
Use the **id** attribute to give the target element a unique name:

```
<h2 id="startH">H</h2>
```

Step 2: Link to the target (#).
The **#** symbol before the name indicates that the link goes to a fragment:

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

Targeting Browser Windows

The **target** attribute in the a tag tells the browser the name of the window in which you want the linked document to open:

```
<a href="recipes.html" target="_blank">Recipe book</a>
```

target="_blank"
Always opens a new browser window.

target="name"
Opens a new window with that name and then opens all subsequent targeted links with that name in the same window.

It may also open in an embedded frame (iframe) with that name.

Mail Links

Use the “mailto” protocol to make a linked email address open in a mail program:

```
<a href="mailto:wonderwoman@example.com">Email
WonderWoman</a> (wonderwoman@example.com)
```

NOTE: Most browsers are configured to open the computer’s primary email program, but this may not work for some users. Be sure the email address is included on the page and use the `mailto` link as progressive enhancement.

3

Block Elements

OVERVIEW

- General block elements
- Breaks
- Lists
- Page organizing elements
- Inline elements
- Generic elements
- ARIA introduction
- Escaping characters

Markup Tips

- It is important to mark up content semantically, in a way that accurately describes the content's meaning or function.
- This ensures your content is accessible in the widest range of viewing environments:
  - Desktop and mobile browsers
  - Assistive reading devices
  - Search engine indexers

Paragraphs

** **

Paragraphs are the most rudimentary elements in a text document.

```
<p>Serif typefaces have small slabs at the ends of
letter strokes. In general, serif fonts can make large
amounts of text easier to read.</p>

<p>Sans-serif fonts do not have serif slabs; their
strokes are square on the end. Helvetica and Arial are
examples of sans-serif fonts. In general, sans-serif
fonts appear sleeker and more modern.</p>
```

Headings

<h#> </h#>

There are six levels of headings (**h1** to **h6**).

```
<h1>Top-Level Heading</h1>
<p>This is a regular paragraph that will display at the browser's
default font size and weight for comparison.</p>

<h2>Second-Level Heading</h2>
<h3>Third-Level heading</h3>
<p>This is another paragraph for comparison. Of course, you can
change the presentation of all of these elements with your own style
sheets.</p>

<h4>Fourth Level Heading</h4>
<h5>Fifth Level Heading</h5>
<h6>Sixth-Level Heading</h6>
<p>This is another paragraph to show the default relationship of
headings to body paragraphs. Of course, you can change the
presentation of all of these elements with your own style sheets.</p>
```

Headings (cont'd)

h1

Top-Level Heading

This is a regular paragraph that will display at the browser's default font size and weight for comparison.

h2

Second-Level Heading

h3

Third-Level heading

This is another paragraph for comparison. Of course, you can change the presentation of all of these elements with your own style sheets.

h4

Fourth Level Heading

h5

Fifth Level Heading

h6

Sixth-Level Heading

This is another paragraph to show the default relationship of headings to body paragraphs. Of course, you can change the presentation of all of these elements with your own style sheets.

Headings (cont'd)

- Used to create the document outline.
- Help with accessibility and search engine indexing.
- Recommended to start with **`h1`** and add subsequent levels in logical order.
- Don't choose headings based on how they look; use a style sheet to change them.

Long Quotations (blockquotes)

<blockquote> </blockquote>

```
<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>
```

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.

Preformatted Text

<pre> </pre>

Preformatted text preserves white space when it is important for conveying meaning. By default, **pre** text displays in a constant-width font, such as Courier.

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

```
This is                 an              example of
      text with a           lot of
                          curious
                          whitespace.
```

Line Breaks

The empty **br** element inserts a line break.

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

So much depends
upon

a red wheel
barrow

Thematic Breaks (Horizontal Rules)

<hr>

Indicates one topic has completed and another one is beginning. Browsers display a horizontal rule (line) in its place:

```
<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>
```

Times

Description and history of the Times typeface.

Georgia

Description and history of the Georgia typeface.

Lists

There are three types of lists in HTML:

- Unordered lists
- Ordered lists
- Description lists

Unordered Lists

In unordered lists items may appear in any order (examples, names, options, etc.). Most lists fall into this category.

`<ul> </ul>`	Defines the whole list
`<li> </li>`	Defines each list item

Unordered Lists (cont'd)

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

- Serif
- Sans-serif
- Script
- Display
- Dingbats

You can change the appearance of the list dramatically with style sheet rules.

Serif
Sans-serif
Script
Display
Dingbats

SERIF
SANS-SERIF
SCRIPT
DISPLAY
DINGBATS

Serif | Sans-serif | Script | Display | Dingbats

SERIF | SANS-SERIF | SCRIPT | DISPLAY | DINGBATS

Ordered Lists

In ordered lists items occur in a particular order, such as step-by-step instructions or driving directions.

`<ol> </ol>` Defines the whole list

`<li> </li>` Defines each list item

Ordered Lists (cont'd.)

```
<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)

Description Lists

Description lists are used for any type of **name/value pairs**, such as terms/definitions, questions/answers, etc.

`<dl> </dl>`	Defines the whole list
`<dt> </dt>`	Defines a name, such as a term
`<dd> </dd>`	Defines a value, such as a definition

Description Lists (cont'd)

```
<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>
</dl>
```

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.

Page Organizing Elements

HTML5 introduced elements that give meaning to the typical sections of a web page:

- `main`
- `header`
- `footer`
- `section`
- `article`
- `aside`
- `nav`

Main Content

<main> </main>

- Identifies the primary content of a page or application
- Helps users with screen readers get to the main content of the page
- Requires JavaScript workaround in Internet Explorer

```
<body>
<header>…</header>
<main>
  <h1>Humanist Sans Serif</h1>
  …content continues…
</main>
</body>
```

Headers and Footers

```
<header> </header>
<footer> </footer>
```

`header` identifies the introductory material that comes at the beginning of a page, section, or article (logo, title, navigation, etc.).

`footer` indicates the type of information that comes at the end of a page, section, or article (author, copyright, etc.)

Headers and Footers (cont'd)

```
<article>
  <header>
    <h1>More about WOFF</h1>
    <p>by Jennifer Robbins, <timedatetime="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>
```

Sections

<section> </section>

section identifies thematic section of a page or an article. It can be used to divide up a whole page or a single 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>
```

Articles

<article> </article>

article is used for self-contained works that could stand alone or be used in a different context (such as syndication).

Useful for magazine/newspaper articles, blog posts, comments, etc.

```
<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>
```

Aside (Sidebar)

<aside> </aside>

aside identifies content that is separate from but tangentially related to the surrounding content (think of it as a sidebar).

```
<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>
```

Navigation

<nav> </nav>

nav identifies the primary navigation for a site or lengthy section or article. It provides more semantic meaning than a simple unordered list.

```
<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>
```

Inline Elements

- Called **text-level semantic elements** in the spec.
- Describe the types of elements that appear in the flow of text.

`a`	`dfn`	`mark`	`s`
`em`	`code`	`time`	`u`
`strong`	`var`	`data`	`small`
`q`	`samp`	`ins/del`	`bdi/bdo`
`abbr`	`kbd`	`b`	`data`
`cite`	`sub/sup`	`i`	`span`

Inline Elements

Emphasis

Text that should be emphasized. Usually displayed in italics.

```
<p><em>Arlo</em> is very smart.</p>
<p>Arlo is <em>very</em> smart.</p>
```


Text that is important, serious, or urgent. Usually displayed in bold.

```
<p>When returning the car, <strong>drop the keys in
the red box by the front desk</strong>.</p>
```

TIP: Use these elements semantically, not to achieve font styles.
Think of how it would be read with a screen reader.

Inline Elements

Short Quotations

<q> </q>

For quoted phrases in the flow of text. Browsers add appropriate quotation marks automatically.

```
<p>Matthew Carter says, <q>Our alphabet
hasn't changed in eons.</q></p>
```

Matthew Carter says, "Our alphabet hasn't changed in eons."

Inline Elements

Abbreviations and Acronyms

<abbr> </abbr>

The **title** attribute provides the long version of a shortened term, which is helpful for search engines and assistive devices.

```
<abbr title="Points">pts.</abbr>

<abbr title="American Type Founders">ATF</abbr>
```

Inline Elements

Superscript and Subscript

****

****

Causes the selected text to display in a smaller size and slightly above (**sup**) or below (**sub**) the baseline.

```
<p>H<sub>2</sub>0</p>
```

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

H_20 $E=MC^2$

Inline Elements

Citations

<cite> </cite>

Identifies a reference to another document.

```
<p>Passages of this article were inspired by <cite>The Complete Manual of Typography</cite> by James Felici.</p>
```

Inline Elements

Defining Terms

<dfn> </dfn>

Identifies the first and defining instance of a word in a document. There is no default rendering, so you need to format them using style sheets.

```
<p><dfn>Script typefaces</dfn> are based on
handwriting.</p>
```

Inline Elements

Code-Related Elements

`<code> </code>`	Code in the flow of text
`<var> </var>`	Variables
`<samp> </samp>`	Program sample
`<kbd> </kbd>`	User-entered keyboard strokes

New Semantic Definitions for Old Presentational Inline Elements

** ** Phrases that need to stand out without added emphasis or importance (***bold***)

** ** Phrases in a different voice or mood than the surrounding text (*italic*)

<s> </s> Text that is incorrect (*strike-through*)

<u> </u> Underlined text, when underlining has semantic purpose

<small> </small> Addendum or side note (*smaller text size*)

Inline Elements

Highlighted Text

<mark> </mark>

For phrases that may be particularly relevant to the reader (for example, when displaying search results):

```
<p> ... PART I. ADMINISTRATION OF THE GOVERNMENT. TITLE
IX. TAXATION. CHAPTER 65C. MASS. <mark>ESTATE
TAX</mark>. Chapter 65C: Sect. 2. Computation of
<mark>estate tax</mark>.</p>
```

... PART I. ADMINISTRATION OF THE GOVERNMENT. TITLE IX. TAXATION. CHAPTER 65C. MASS. ESTATE TAX. Chapter 65C: Sect. 2. Computation of estate tax.

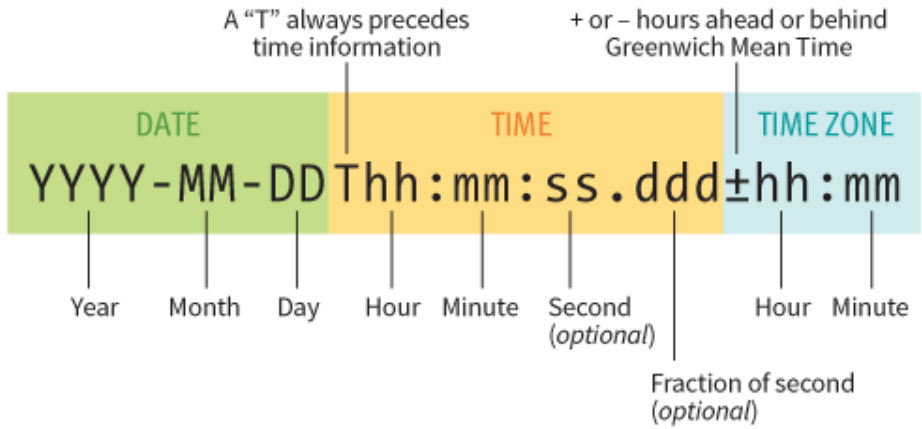
Inline Elements

Dates and Times

<time> </time>

Provides machine-readable equivalents for dates and times. The **datetime** attribute specifies the date/time information in a standardized time format:

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



Inline Elements

Machine-Readable Information

`<data> </data>`

Helps computers make sense of content.

The **`value`** attribute provides the machine-readable information.

```
<data value="12">Twelve</data>

<data value="978-1-449-39319-9">CSS: The
Definitive Guide</data>
```

Inserted and Deleted Content

<ins> </ins>

<del> </del>

Markup for edits indicating parts of a document that have been inserted or deleted:

```
Chief Executive Officer: <del
title="retired">Peter Pan</del><ins>Pippi
Longstocking</ins>
```

Generic Elements

`<div> </div>`

Indicates division of content (generally block-level)

`<span> </span>`

Indicates a word or phrase

- Generic elements are given semantic meaning with the **`id`** and **`class`** attributes.
- They are useful for creating “hooks” for scripts and style rules.

Div Example

Use the **div** element to create a logical grouping of content or elements on the page.

It indicates that they belong together in some sort of conceptual unit or should be treated as a unit by CSS or JavaScript.

```
<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>
```

Span Example

Use the **span** element for text and other inline elements for which no existing inline element currently exists.

In this example, a **span** is used to add semantic meaning to telephone numbers:

```
<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>
```

id and class Attributes

`id`
Assigns a unique identifier to the element.

`class`
Classifies elements into a conceptual group.

Use the **id** attribute to identify.
Use the **class** attribute to classify.

NOTE: **id** and **class** can be used with *all* HTML elements.

The id Attribute

The value of an **id** attribute must be used only once in a document.

Here it identifies a listing for a particular book by its ISBN:

```
<div id="ISBN0321127307">
  <img src="felici-cover.gif" alt="">
  <p><cite>The Complete Manual of
Typography</cite>, James Felici</p>
  <p>A combination of type history …</p>
</div>
```

Here it identifies a particular section of a document:

```
<section id="news">
  <!-- news items here -->
</section>
```

The class Attribute

A **class** value may be used by multiple elements to put them in conceptual groups for scripting or styling.

Here several book listings are classified as a “listing”:

```
<div id="ISBN0321127307" class="listing">
 …
</div>

<div id="ISBN0881792063" class="listing">
…
</div>
```

An element may belong to more than one class. Separate class values with character spaces:

```
<div id="ISBN0321127307" class="listing book
nonfiction">
```

Brief ARIA Introduction

ARIA (Accessible Rich Internet Applications) is a standardized set of attributes for making pages easier to navigate and use with assistive devices.

ARIA defines **roles**, **states**, and **properties** that developers can add to markup and scripts to provide richer information.

www.w3.org/TR/html-aria

ARIA Roles

Roles describe or clarify an element's function in the document.

Examples: `alert`, `button`, `dialog`, `slider`, and `menubar`

```
<div id="status" role="alert">You
are no longer connected to the
server.</div>
```

ARIA States and Properties

- ARIA defines a long list of **states** and **properties** that apply to interactive elements and dynamic content.
- Properties values are likely to be stable (example: `aria-labelledby`).
- States have values that are likely to change as the user interacts with the content (example: `aria-selected`).

Escaping Characters

Escaping a character means representing it by its named or numeric **character entity** in the source.

- Some characters must be escaped because they will be mistaken for code (example: the < character would be parsed as the start of an HTML tag).
- Some characters are invisible or just easier to escape than find on the keyboard.

Character Entity References

Character entities always begin with **&** and end with **;**.

Named entities
Use a predefined name for the character
(example: `<` for the less-than symbol **<**)

Numeric entities
Use an assigned numeric value that corresponds to its position in a coded character set, such as UTF-8
(example: `<` for the less-than symbol **<**).

A complete list of HTML named entities and their Unicode code-points is at www.w3.org/TR/html5/syntax.html#named-character-references.

Escaping HTML Syntax Characters

Always escape **<**, **>**, and **&** characters in content.
Escape **"** and **'** when they are in attribute values.

Character	Description	Entity name	Decimal	Hexadecimal
<	Less-than symbol	<		
>	Greater-than symbol	>	>	>
"	Quote	"		"
'	Apostrophe	'	'	'
&	Ampersand	&	&	&

(Additional non-required character entities are listed in Chapter 5.)

4

Adding Images

OVERVIEW

- Web image formats
- The `img` element
- Attributes for the img element
- Adding an SVG to a page

Web Image Formats

Image formats appropriate for web pages:

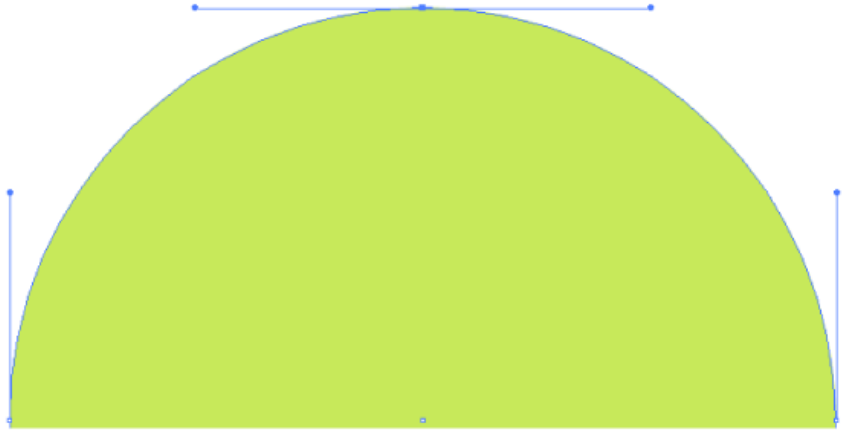
- PNG
- JPEG
- GIF
- SVG
- WebP (up and coming, not yet widely supported)

Bitmapped vs. Vector Formats

Bitmapped images are made up of a grid of colored pixels.

PNG, JPEG, GIF, and WebP are bitmapped formats.

Vector images contain paths that are defined mathematically.



SVG is a vector format.

The img Element

```
<img src="" alt="">
```

- Embed images on the page with the empty **`img`** element.
- The **`src`** and **`alt`** attributes are required.
- **`img`** can be used for PNG, JPEG, GIF, and SVG.

NOTE: Special markup is recommended for experimenting with cutting-edge image formats like WebP.

The img Element (cont'd)

The **img** element tells the browser to make a server request for the image and display it in its place:

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

This summer, try making pizza  on your grill.

The src attribute

```
<img src="/images/icon/next.png" alt="Next">
```

- The **value** is the location (URL) of the image file.
- Use the relative pathname conventions to point to the image on the server.

PERFORMANCE TIP: Take advantage of **caching** (temporary storage). For the same image used repeatedly, using the same pathname reduces the number of calls to the server.

The alt Attribute

```
<p> If you're <img src="/images/smiley.png" alt="happy"> and
you know it, clap your hands.</p>
```

- The **`alt`** attribute provides **alternative text** for those who are not able to see it.
- It is **required** for every **`img`** element in order for the HTML to be valid.
- It is used by screen readers, search engines, and graphical browsers when the image fails to load.

With image displayed

If you're  and you know it clap your hands.

Firefox

If you're happy and you know it clap your hands.

Chrome (Mac & Windows)

If you're happy and you know it clap your hands.

MS Edge (Windows)

If you're happy and you know it clap your hands.

Alternative Text Tips

- Alternative text (alt text) should convey the same information and function as the image.
- If the image is purely decorative and shouldn't be read aloud, include the **`alt`** attribute with an empty value:

```
<img src="flowers.svg" alt="">
```

- Consider what the alt text would sound like when read aloud by a screen reader. Is it helpful or a hindrance?
- When an image is used as a link, the alt text serves as the linked text. Write what you'd want the link to say, don't just describe the image.
- Avoid starting alt text with "An image of" or "Graphic of".

width and height Attributes

```
<img src="flowers.svg" alt="" width="120" height="160">
```

- The **width** and **height** attributes set the dimensions of the image on the page in number of pixels.
- They help the browser maintain space for the image in the layout while the files load.
- Don't use **width** and **height** attributes if you are sizing the image with style sheets or if the size changes in a responsive layout.
- If the attribute values do not match the dimensions of the image, the image will resize and may be distorted or blurry.

Adding SVG Images

SVG images are unique:

- SVG is a vector format, made up of shapes and paths.
- Those shapes and paths are described in a text file using the **Scalable Vector Graphic** markup language.
- The elements in an SVG can be styled with CSS and scripted for interactivity.
- Because they are vector, SVGs can resize with no loss of quality.

SVG Example

Compare the SVG source to the image. The **`svg`** element contains a rectangle (**`rect`** element) and a **`text`** element:

```
<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>
```

Adding SVG to a Page

There are several ways to add an SVG image to a web page:

- `<img>` element
- `<object>` element
- `<svg>` element directly in HTML (inline SVG)

Adding SVG with the img Element

You can add an *.svg* file to the page with the **`img`** element:

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

PROS:

- Easy and familiar
- Universally supported

CONS:

- Can’t manipulate the SVG with scripts or styles.
- The SVG can’t contain any external resources such as images or fonts.

Embedding SVGs with object

The content of the **`object`** element is a fallback text or image that displays if the SVG is not supported:

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

PROS:

- SVG can be scripted and use eternal files (images and fonts).

CONS:

- You can’t change styles with the same CSS used for the document.
- May be buggy in some browsers.

Inline SVGs with the svg Element

You can paste the content of the SVG text file directly into the HTML source. This is called using the SVG **inline**.

PROS:

- Can take full advantage of scripting and styling the SVG because the elements in the SVG are part of the DOM for the document.

CONS:

- Code can get extremely long and unwieldy.
- Harder to maintain images embedded in the source
- Can’t take advantage of caching repeated images
- Not universally supported

Example of an Inline SVG

This summer, try making pizza  on your grill.

```
<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>
```