# Web Design and Programming

Week 2

25 April 2024

Instructor: Dr. Peeraya Sripian

# Course schedule

| Week | Date | Topic |
|---|---|---|
| 1 | 4/18 | Intro to WWW, Intro to HTML |
| 2 | 4/25 | CSS Fundamental |
| | 5/2 | Holiday (GW) |
| 3 | 5/9 | CSS and Bootstrap |
| 4 | 5/16 | Work on midterm project |
| 5 | **COIL 22 MAY 18:00-19:30 (counted as 1 class, replacing 23 May)** | |
| 6 | 5/30 | Midterm project presentation week |
| 7 | 6/6 | PHP fundamentals + Installation XAMPP |
| 8 | 6/13 | PHP fundamentals 2 |
| 9 | 6/20 | mySQL fundamentals |
| 10 | 6/27 | Assessing MySQL using PHP, MVC pattern + Intro of Final project |
| 11 | 7/4 | Cookies, sessions, and authentication + Proposal of final project |
| 12 | 7/11 | Javascript and PHP validation |
| 13 | 7/18 | Final project development |
| 14 | 7/25 | Final project presentation |

# This week’s learning objective

- Basic HTML for structuring the webpage
  - Mark up text
  - Links
  - Images
  - Tables

# Today’s outline

- Review from last time
- HTML for structure
  - Marking up text

Break 10 mins

  - Adding links
  - Adding images
  - Table markups

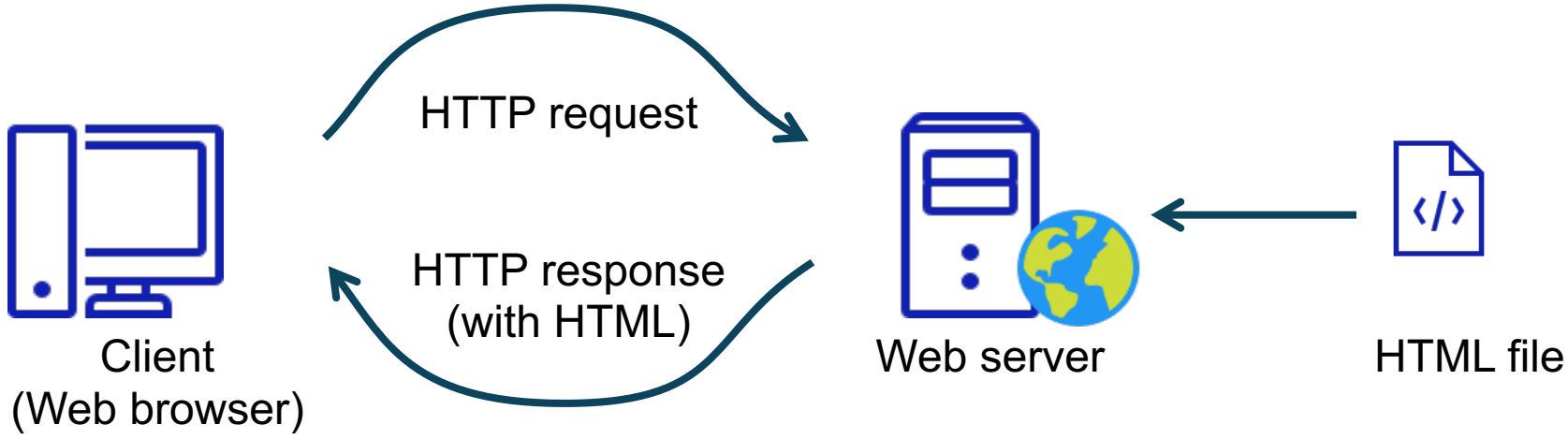
Break 10 mins

- Midterm project explanation
- In class activity

# Internet vs. WWW in summary

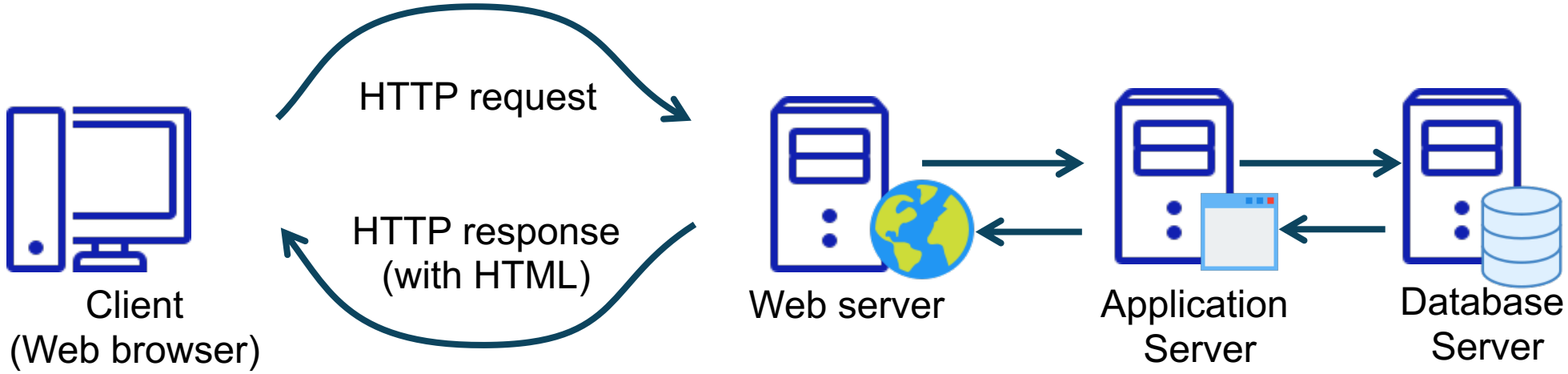
- Internet is a worldwide set of computer networks for personal, commercial, academic, and government use.
- WWW is a subset of the internet containing a set of web pages and sites.
- IP is used for addressing of internet communication, each computer on the internet has IP address.
- Web emails file transfer are layered on top of TCP
- URL map to IP address through a service called DNS
- HTTP allows a browser to request a web document
- Web programming involves many languages: HTML, CSS, PHP, Javascript, XML, and SQL

# How static web pages are processed




- A static webpage is an HTML document stored on a web server and does not change
- Usually with filenames .htm .html
- When user request a static webpage, the browser send HTTP request include the filename
- Web server retrieve HTML from webpage and send it back to browser as part of HTTP response
- Browser then renders the HTML into a webpage and display

# How dynamic web pages are processed




- A dynamic webpage is a webpage that is generated by a server-side program or script.
- A web server would looks up the **extension** of the requested file to find out which **application server** should process the request
- Application server runs the specified script. Often uses the data get from web browser to get the appropriate data from a **database server**.
- When application server finishes processing the data, it generates the **HTML** for a web page and returns to web server, then return to web browser as part of HTTP response.

# Structure of a minimal HTML page

```
<!DOCTYPE html>
<html>

  <head>
    information about the page
  </head>

  <body>
    page contents
  </body>

</html>
```

Header

Body

- the **header** describes the page
- the **body** contains the page's contents
- an **HTML** page is saved into a file ending with extension .html
- **DOCTYPE** tag tells browser to interpret our page's code as HTML5, the

# About HTML codes

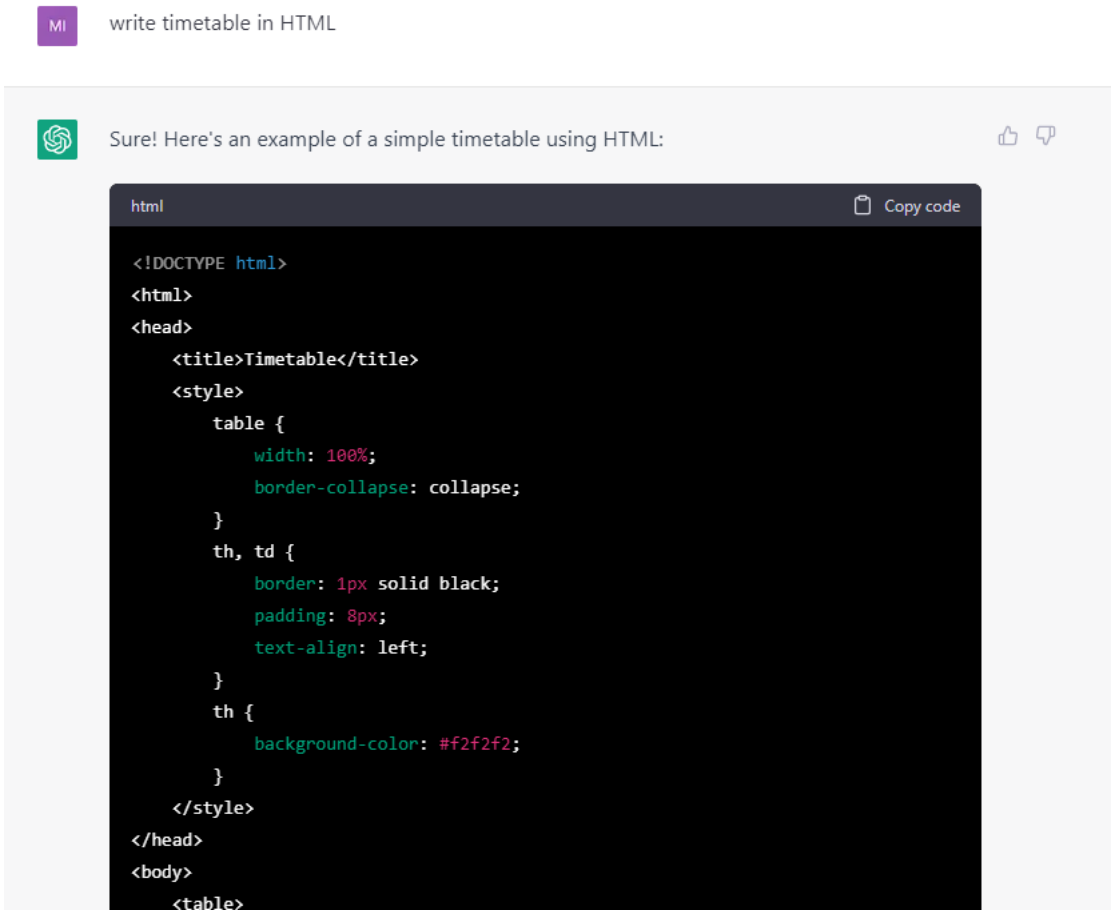
- You can find all necessary HTML codes and explanation from w3schools.com

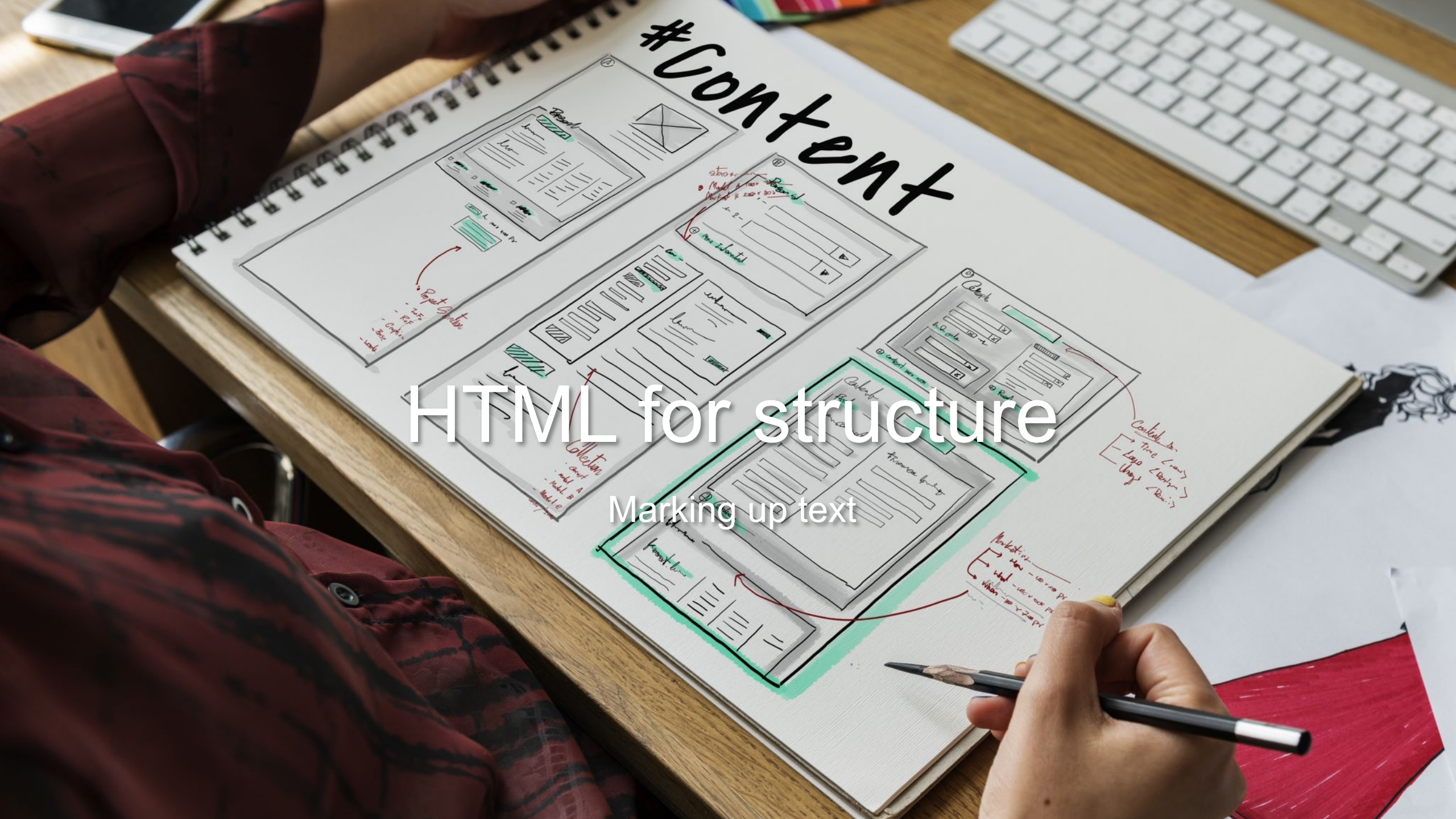
- This lecture will provide insight to the necessary HTML structure, often used tags only.

- Please make a good use of w3schools and ChatGPT (yes, of course) when you want to create the website

# ChatGPT and website

- ChatGPT can be a valuable resource
- However, the most important is *“how you get to the result”*, not the result itself
- So the process of learning is what you should achieve through this class
- You can use it to *help check your code* for errors, to explain the difficult code for your *better understanding*
- Failure to explain your own source code will result in receiving *zero* for the assignment






# HTML for structure

Marking up text

# 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

 Use often

# Paragraphs

The easy way to try your code: https://jsfiddle.net/

`<p>` `</p>`

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>
```

Use often

# 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>
```

Output in the browser

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.

Skip explanation

# 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>
```

Output in the browser

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.

Skip explanation

# 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>
```

Output in the browser

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

 Use often

# Line Breaks

## `<br>`

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

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

Output in the browser

So much depends
upon

a red wheel
barrow

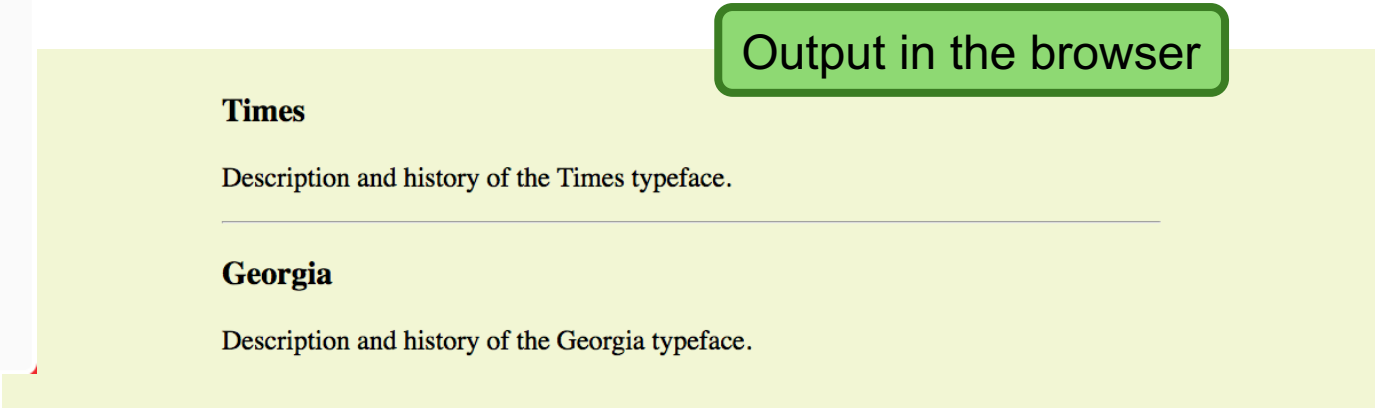
# Thematic Breaks (Horizontal Rules)

**<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>
```

Output in the browser



# Lists

There are three types of lists in HTML:

- Unordered lists
- Ordered lists
- Description lists

Use often

# 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

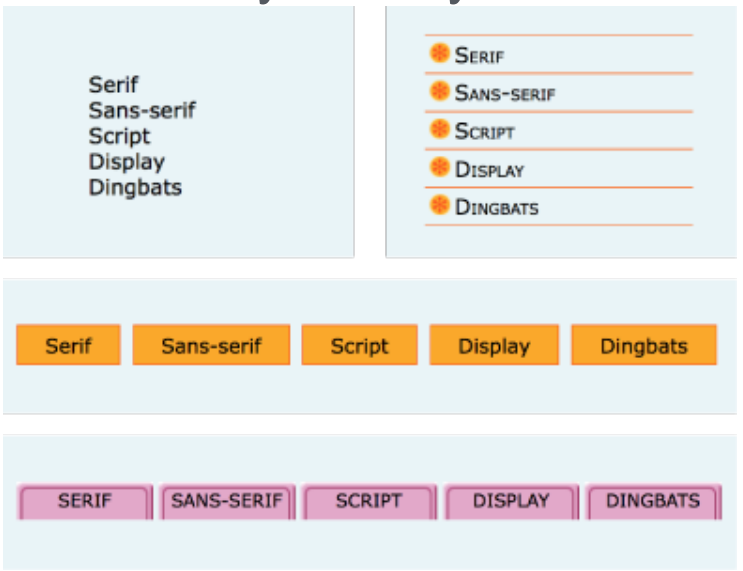
`<li> </li>` Defines each list item

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

Output in the browser

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

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



Use often

# 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

```
<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>
```

Output in the browser

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

Skip explanation

# 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

```
<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>
```

Output in the browser

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`

 Use often

# 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>
```

 Use often

# 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.)

```
<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>
```

 Use often

# 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>
```

 Use often

# 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>
```

Use often

# 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>
```

Use often

# 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` |

 Use often

Inline Elements

# Emphasis

`<em>` `</em>`

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>
```

`<strong>` `</strong>`

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.

 Use often

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."

Output in the browser

Skip explanation

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

**<sup> </sup>**

**<sub> </sub>**

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>
```

$H_20$ $E=MC^2$

Output in the browser

Skip explanation

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>
```

Output in the browser

Passages of this article were inspired by *The Complete Manual of Typography* by James Felici.

Skip explanation

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

`<b> </b>` Phrases that need to stand out without added emphasis or importance (bold)

`<i> </i>` 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)

Skip explanation

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>
```

Output in the browser

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

Skip explanation

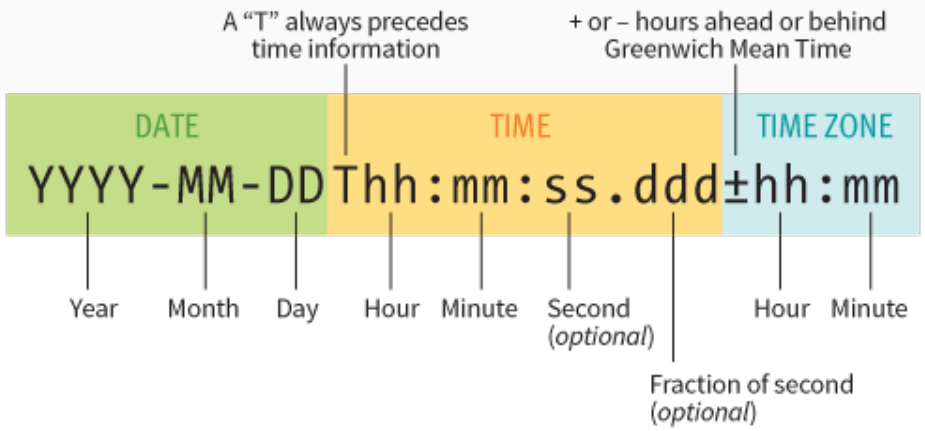
Inline Elements

# 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>
```



Skip explanation

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>
```

Inline Elements

# 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>
```

 Use often

# 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>
```

 Use often

# 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">
```

# Improving Accessibility with ARIA (WAI-ARIA)

www.w3.org/TR/html-aria

- **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.
- Considering that your web users may be listening to the content on the page read by the screen reader, or use other non-mouse devices to navigate your page
- HTML elements such as title, headings, images are semantically meaningful in itself
- However, generic elements (div, span) needs some assitance

# 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>
```

```
<nav role="navigation">
<header role="banner">
<main role="main">
<aside role="complementary">
<footer role="contentinfo">
```

# 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: `&#060;` 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 | &lt; | | |
| > | Greater-than symbol | &gt; | &#062; | &#x3E; |
| " | Quote | &quot; |   | &#x22; |
| ' | Apostrophe | &apos; | &#039; | &#x27; |
| & | Ampersand | &amp; | &#038; | &#x26; |

Break 5 min

# HTML for structure

Adding links

Use often

# Adding Links

`<a> </a>`

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

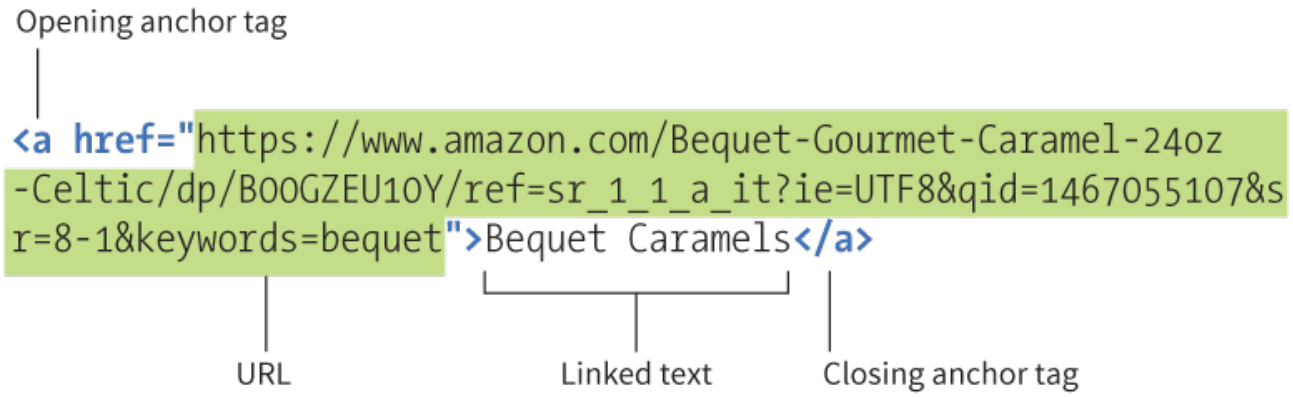
The **`href`** attribute provides the location (URL) of the resource.

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:



# 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>
```

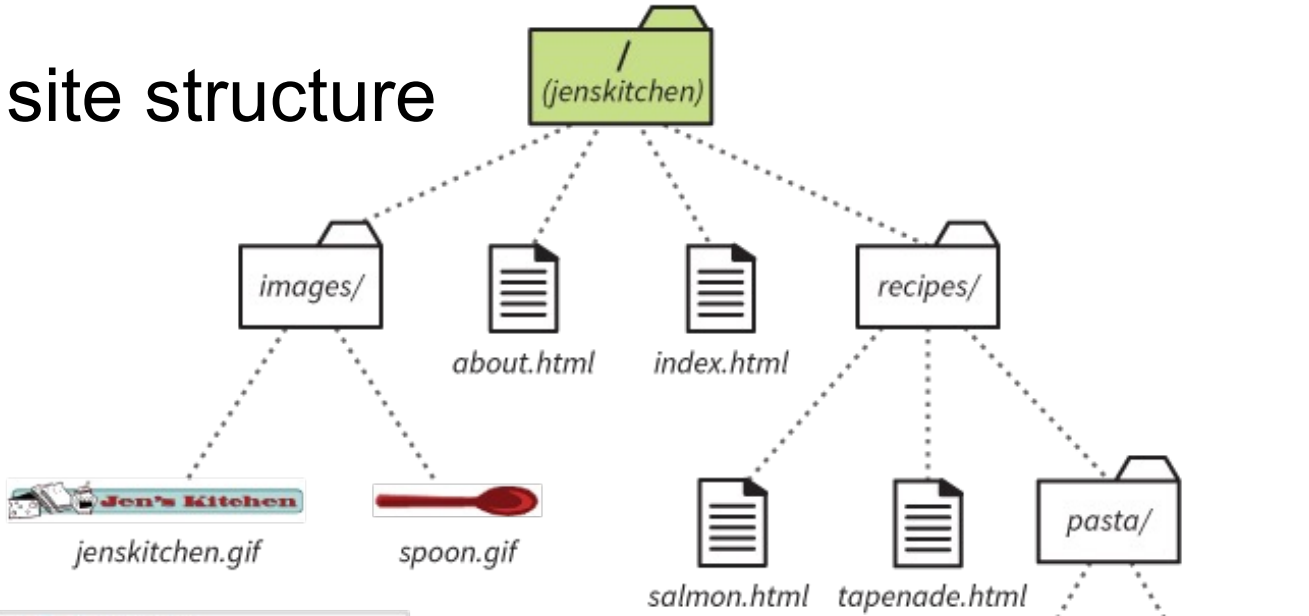
# Linking on Your Own Site

- 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.

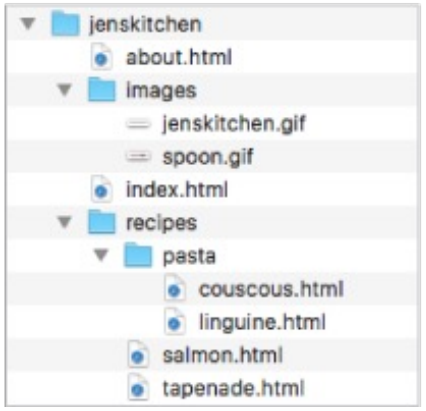
# Example Server Directory Structure

In this section, we will use this site structure

The root directory is called ***jenskitchen***.



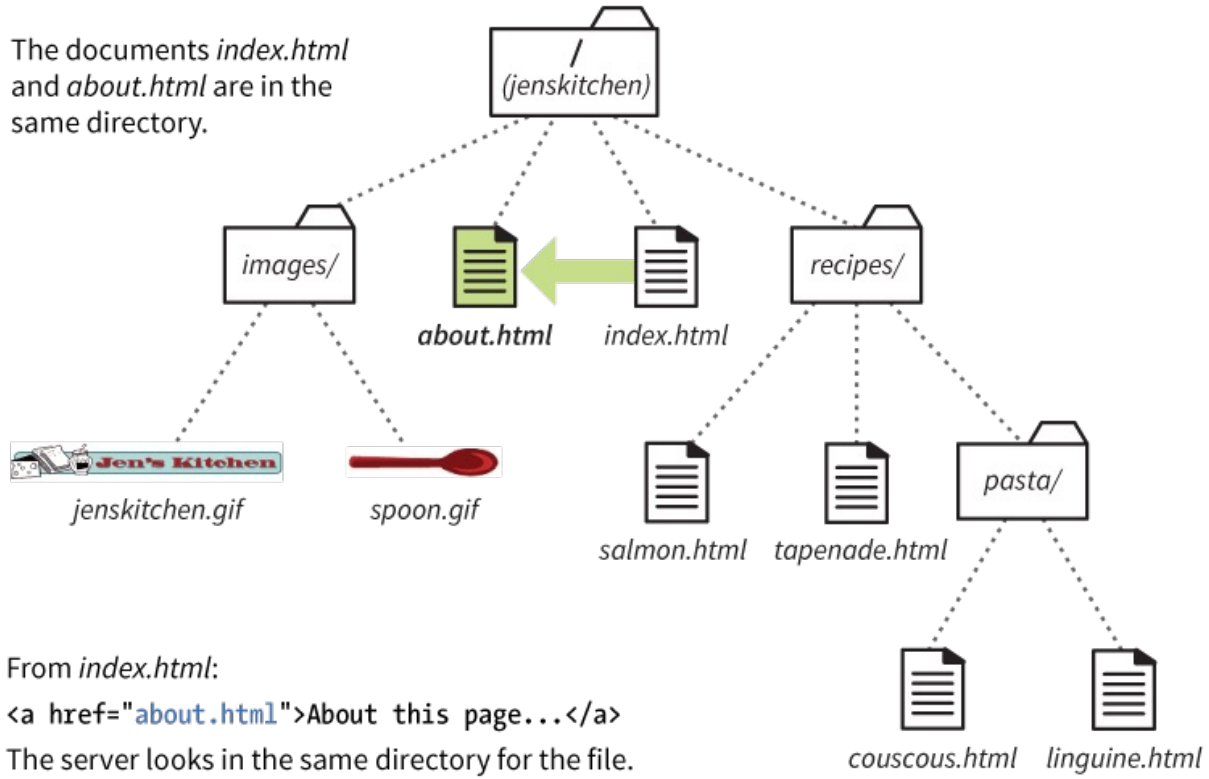

How it looks in the MacOS Finder



# Linking in the Same Directory

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

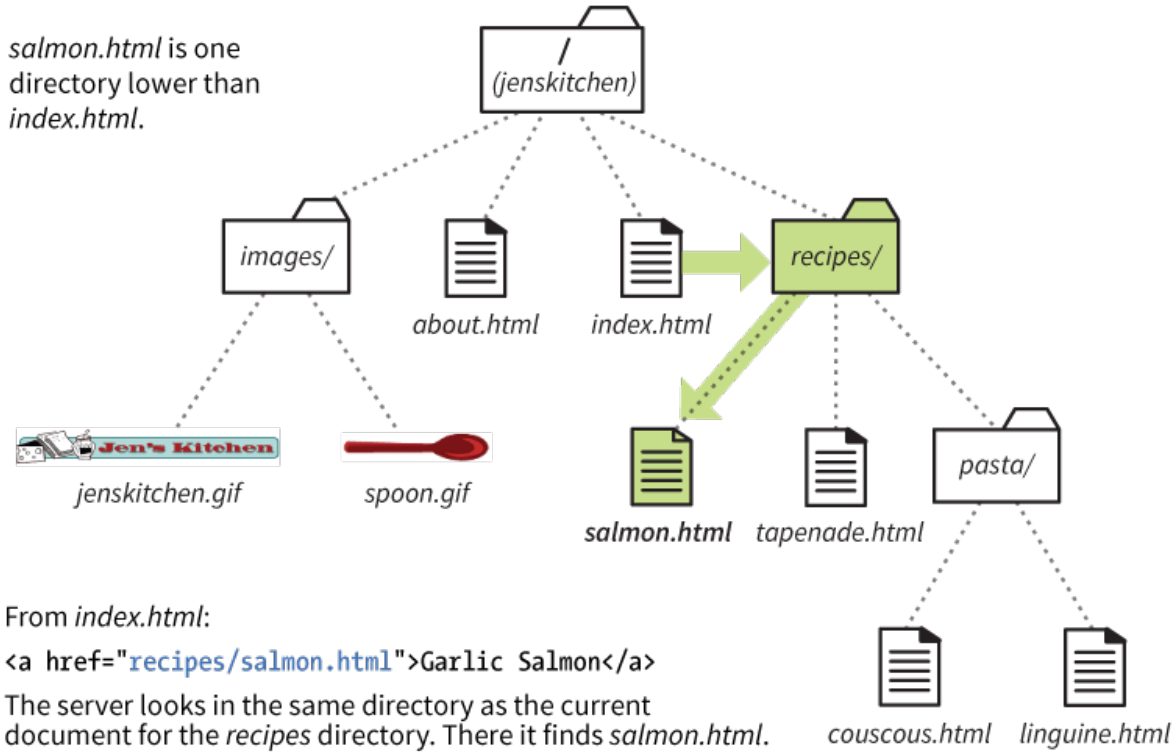
```
href="about.html"
```



# Linking into a Lower Directory

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

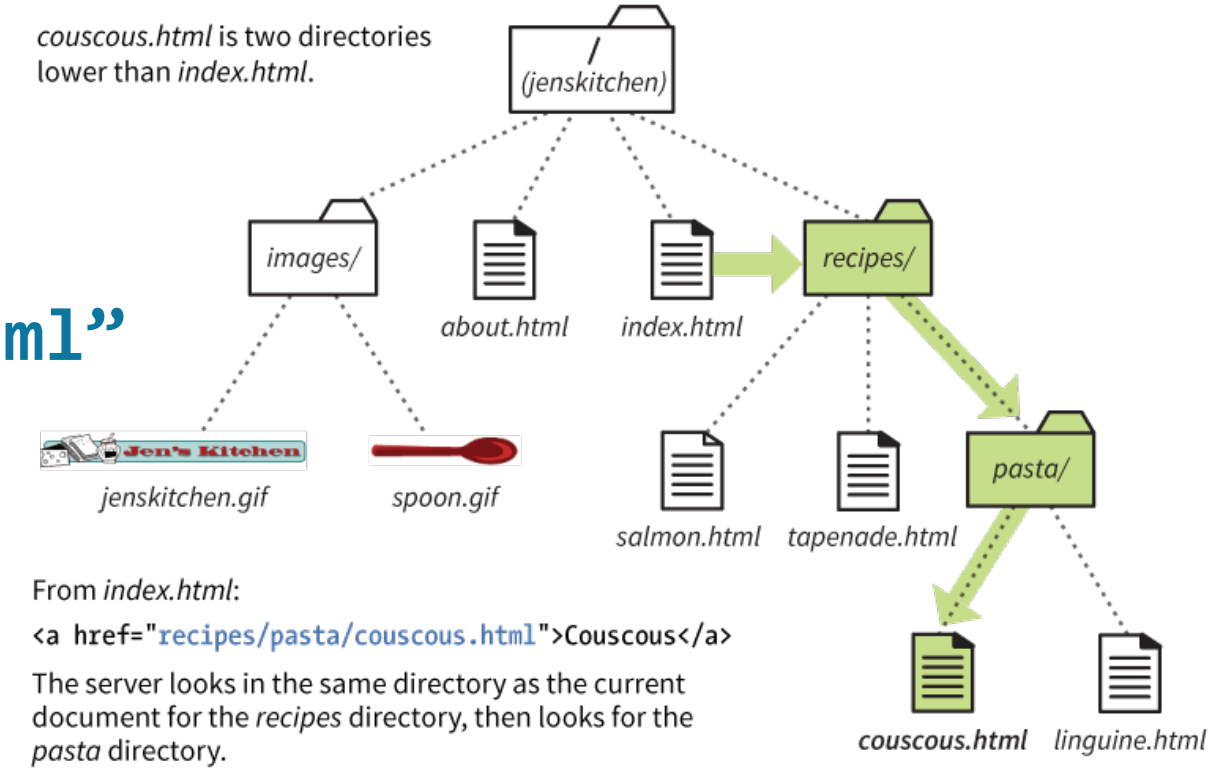
```
href="recipes/salmon.html"
```



# Linking into Two Directories

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

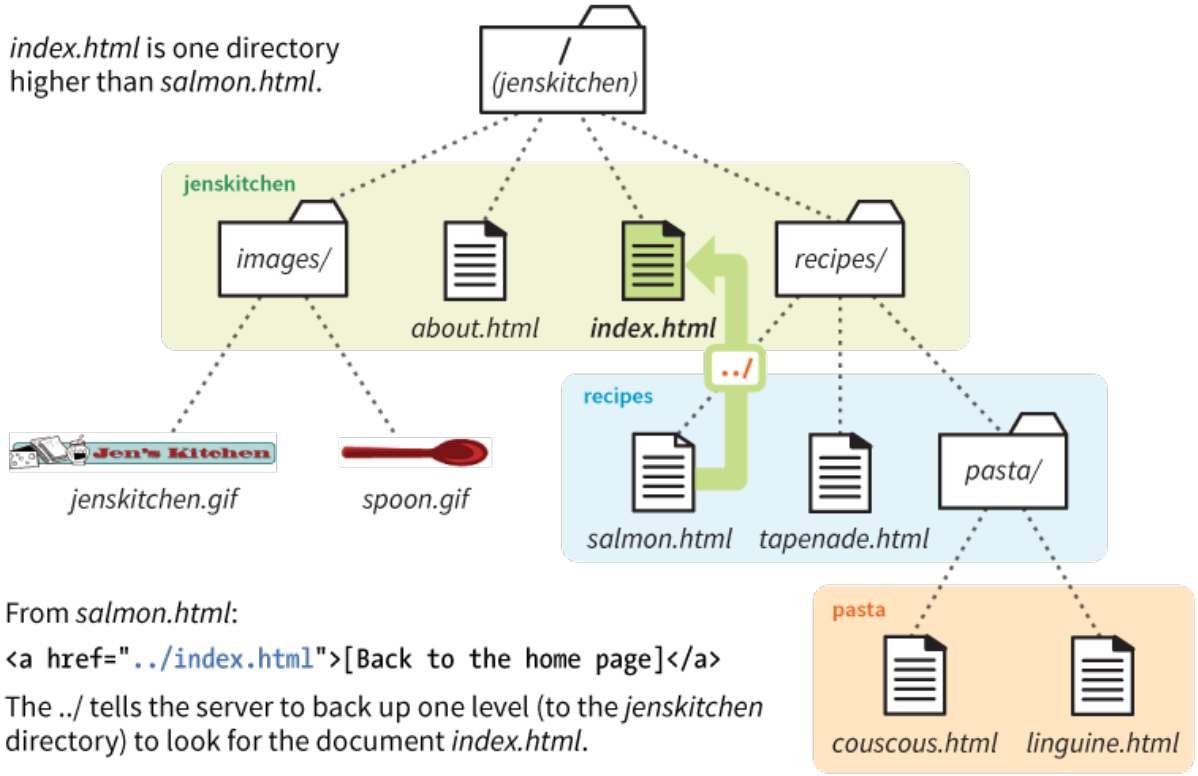
```
href="recipes/pasta/couscous.html"
```



# Linking to a Higher Directory

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

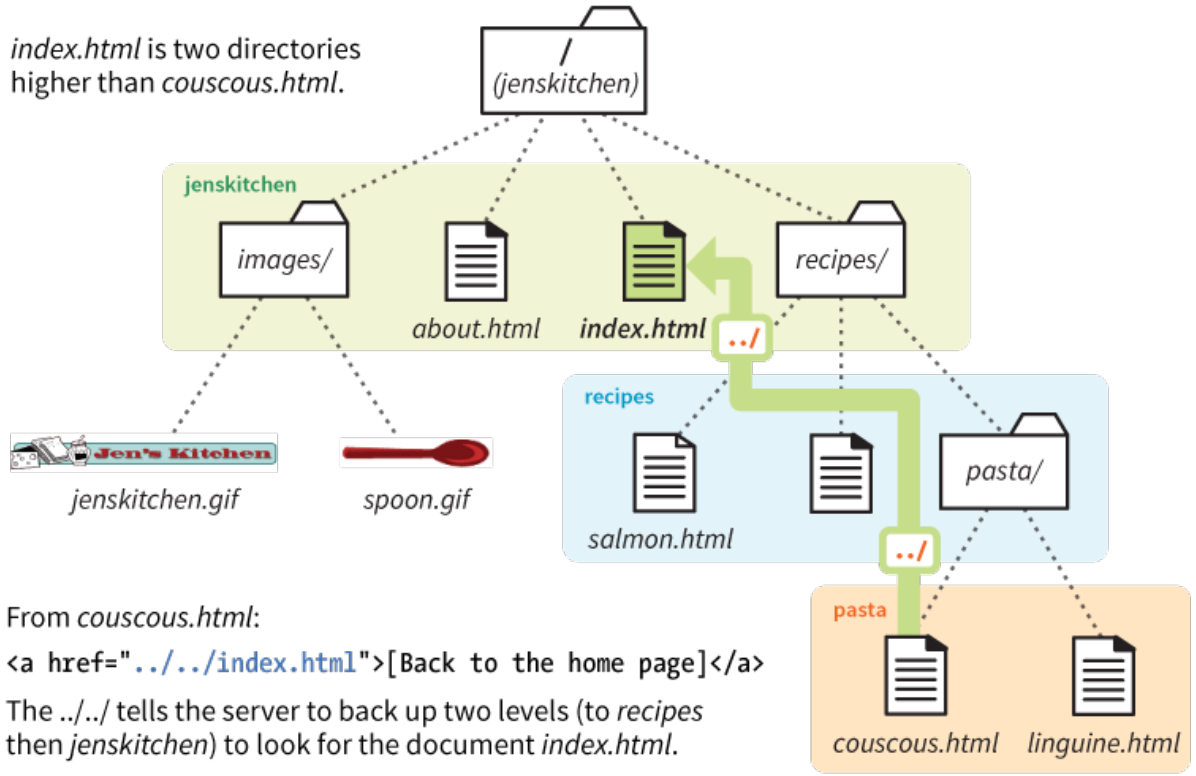
```
href="../index.html"
```



# Linking Up Two Directory Levels

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

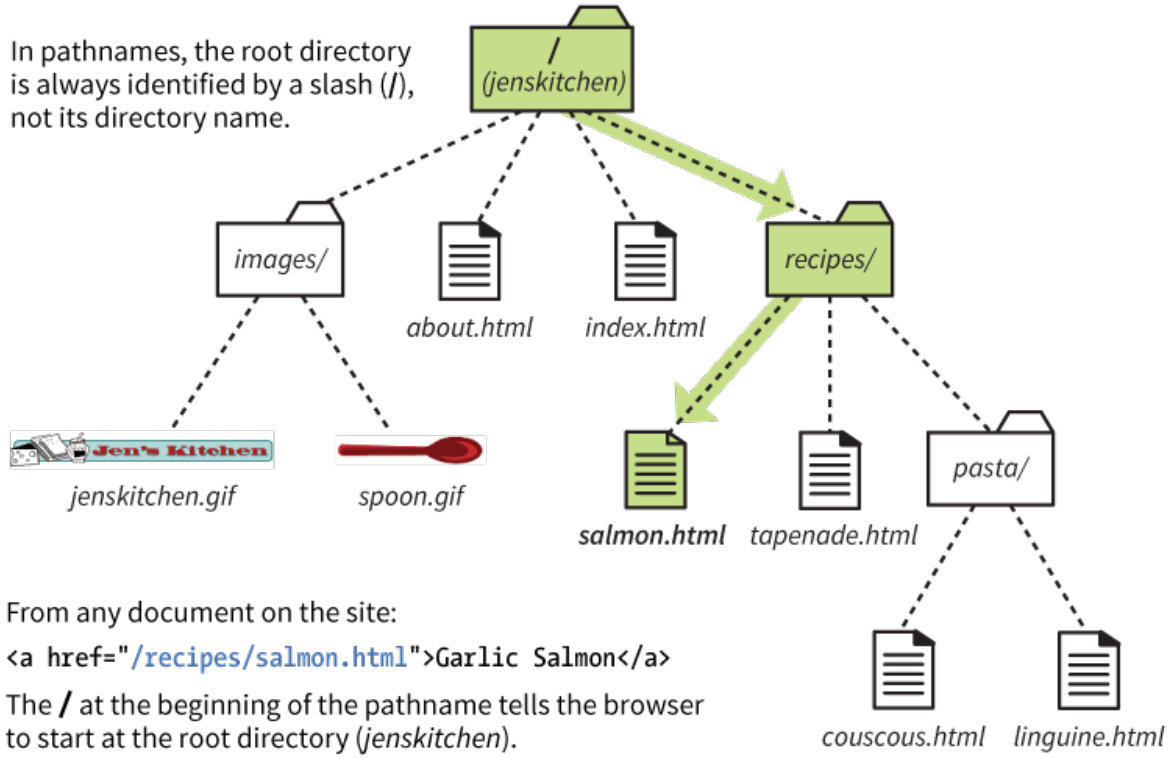
```
href="../../index.html"
```



# 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

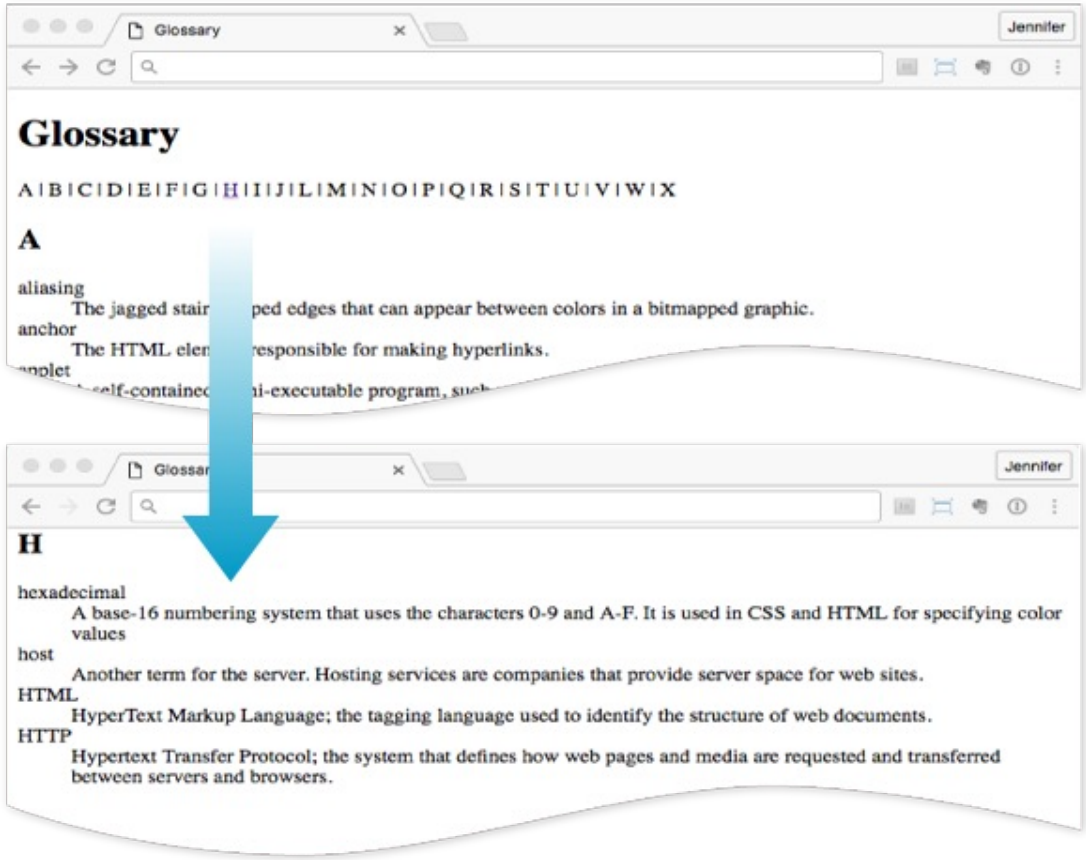
- Relative pathnames are also commonly used to point to image files with the **`src`** attribute:

```
<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.

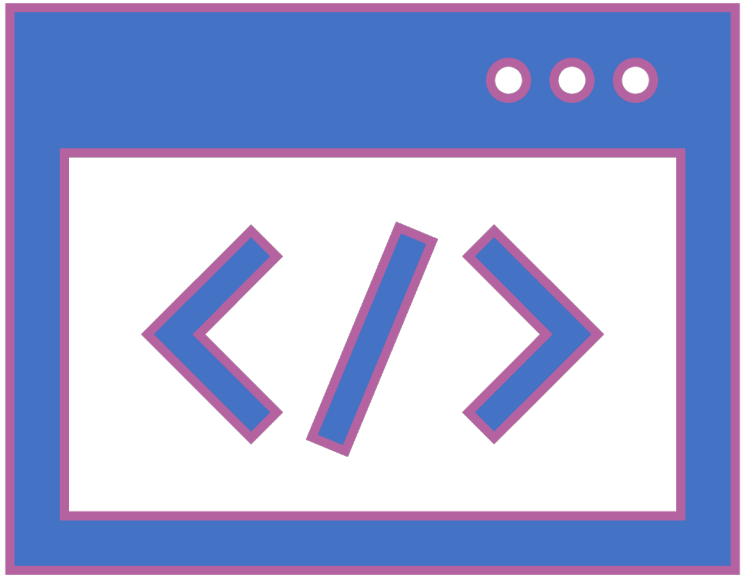
**`target="name"`**
Opens a new window with that name and then opens all subsequent targeted links with that name in the same window.
It may also open in an embedded frame (`iframe`) with that name.

# 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.



# HTML for structure

Adding images

# Web Image Formats

Image formats appropriate for web pages:

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

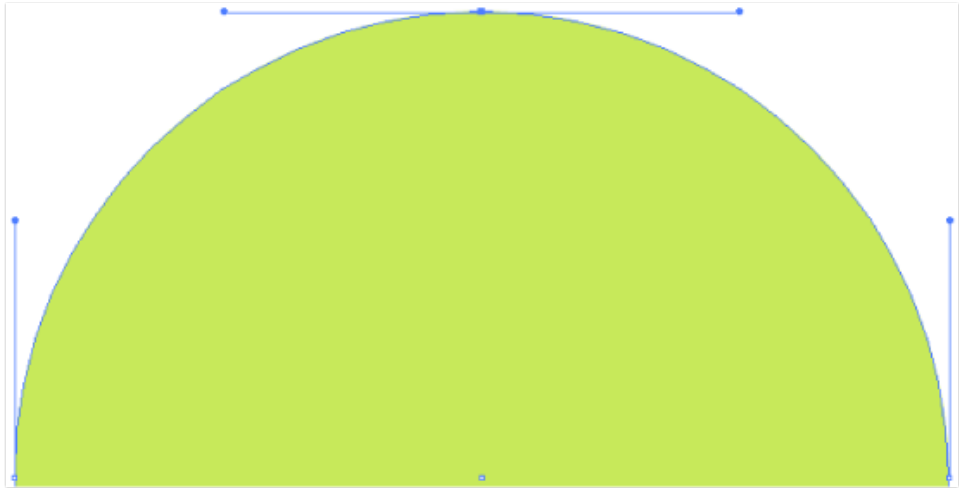
# Bitmapped vs. Vector Formats

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.

Use often

# 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>
```

Output in the browser

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.

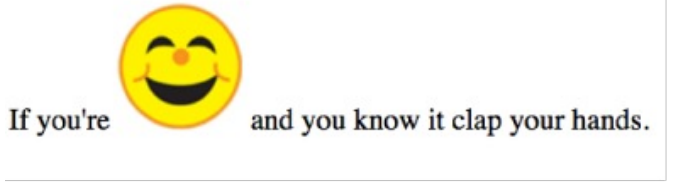
# The alt Attribute

```
<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.

Output in the browser

With image displayed




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>
```

Output in the browser

# 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

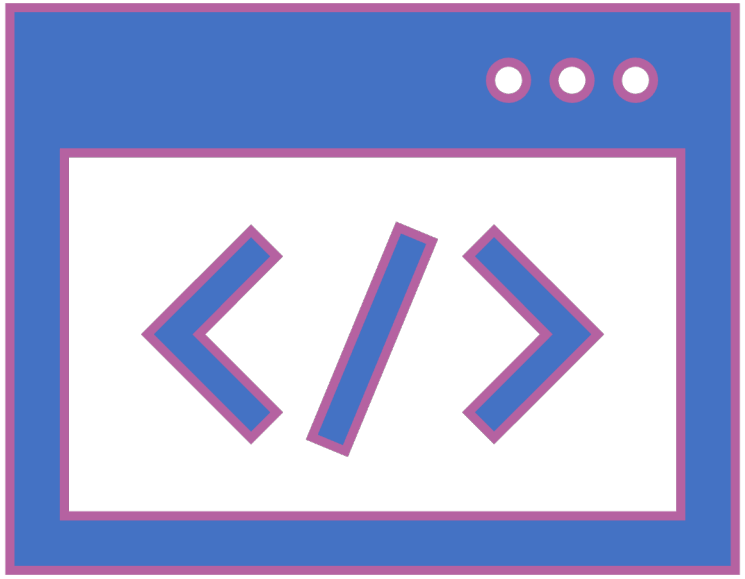
# Example of an Inline SVG

```
<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>
```

Output in the browser

This summer, try making pizza  on your grill.



# HTML for structure

Table markup

# Tabular Data

HTML table markup is for data arranged into rows and columns.

| PM | 7:30 | 8:00 | 8:30 | 9:00 | 9:30 | 10:00 | 10:30 |
|---|---|---|---|---|---|---|---|
| ABC | *The Adventures of Ozzie and Harriet* | The Patty Duke Show | *Gidget* | *The Big Valley* | | Amos Burke — Secret Agent* | |
| CBS | Lost in Space | | *The Beverly Hillbillies* #8 25.9 rating | *Green Acres* #11 24.6 rating | The Dick Van Dyke Show #16 23.6 rating | *The Danny Kaye Show* | |
| NBC | *The Virginian* #25 22.0 rating | | | *Bob Hope Presents the Chrysler Theatre* / *Chrysler Presents a Bob Hope Special* | | *I Spy* | |

*wikipedia.org*

**Providence/Stoughton Line**

Print this Schedule | Providence/Stoughton Line Schedule
Service Alerts | South Station and Back Bay Schedule
Holiday Schedule Information

Direction: Inbound Timing: Weekday Redisplay Time

**PROVIDENCE/STOUGHTON LINE INBOUND : Weekday Effective 11/14/11**

| Train Number | 800 AM | 802 AM | 902 AM | 804 AM | 904 AM | 806 AM | 832 AM | 808 AM | 906 AM | 810 AM | 908 AM | 812 AM | 834 AM | 910 AM | 814 AM | 912 AM | 816 AM | 818 PM | 914 PM | 916 PM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TF Green Airport | | 05:05 | | | | 06:13 | | 06:52 | | 07:15 | | | | | 09:23 | | 11:45 | | | |
| Providence | 05:07 | 05:25 | | 06:07 | | 06:33 | | 07:12 | | 07:35 | | 08:10 | | | 09:43 | | 12:05 | 01:30 | | |
| South Attleboro | 05:17 | 05:35 | | 06:16 | | 06:42 | | 07:22 | | 07:45 | | 08:20 | | | 09:52 | | 12:15 | 01:42 | | |
| Attleboro | 05:27 | 05:45 | | 06:28 | | 06:52 | | 07:32 | | 07:55 | | 08:30 | 09:00 | | 10:02 | | 12:25 | 01:51 | | |
| Mansfield | 05:36 | 05:55 | | 06:38 | | 07:04 | 07:26 | 07:44 | | 08:05 | | 08:38 | 09:09 | | 10:10 | | 12:33 | 01:58 | | |
| Sharon | 05:44 | 06:04 | | 06:48 | | 07:13 | 07:35 | | | 08:14 | | 08:47 | 09:17 | | 10:19 | | 12:42 | 02:06 | | |
| Stoughton | | | 06:28 | | 06:56 | | | | 07:48 | | 08:28 | | | 09:40 | | 10:40 | | | 02:20 | 03:23 |
| Canton Center | | | 06:36 | | 07:04 | | | | 07:57 | | 08:36 | | | 09:49 | | 10:49 | | | 02:27 | |
| Canton Junction | 05:51 | 06:11 | 06:39 | | 07:08 | | 07:41 | | 08:01 | 08:24 | 08:40 | 08:54 | 09:24 | 09:52 | 10:26 | 10:52 | 12:50 | | 02:30 | 03:33 |
| Route 128 | 05:56 | 06:16 | 06:44 | 06:58 | 07:14 | 07:24 | 07:47 | | 08:07 | 08:30 | 08:45 | 08:59 | 09:26 | 09:57 | 10:31 | 10:57 | 12:55 | 02:16 | | 03:38 |
| Hyde Park | 06:01 | 06:21 | 06:49 | | 07:19 | | 07:52 | | 08:13 | 08:36 | 08:49 | 09:04 | | 10:02 | 10:36 | 11:02 | 01:00 | | 02:39 | 03:43 |

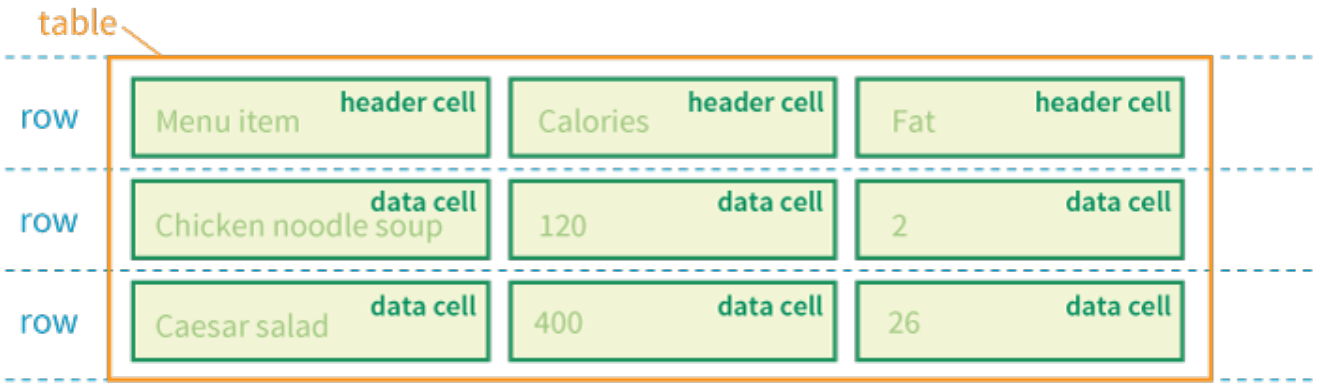
*mbta.org*

# HTML Table Structure

- Tables are made up of cells arranged into rows.

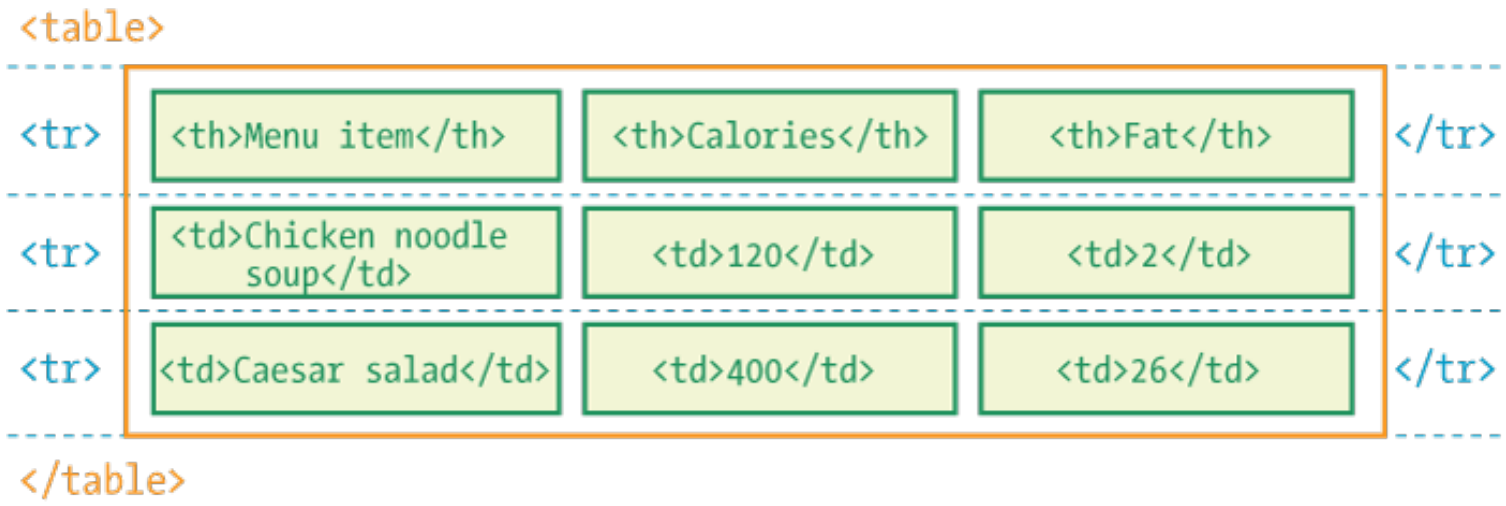
**A simple table**

| Menu item | Calories | Fat (g) |
|---|---|---|
| Chicken noodle soup | 120 | 2 |
| Caesar salad | 400 | 26 |



# HTML Table Structure (cont'd.)

- How it looks using markup (**`table`**, **`tr`**, **`th`**, and **`td`**):




**NOTE:** Columns are implied by the number of cells in each row (not created with column elements).

# HTML Table Structure (cont'd)

The same table written in code:

```
<table>
   <tr>
      <th>Menu item</th>
      <th>Calories</th>
      <th>Fat (g)</th>
   </tr>
   <tr>
      <td>Chicken noodle soup</td>
      <td>120</td>
      <td>2</td>
   </tr>
   <tr>
      <td>Caesar salad</td>
      <td>400</td>
      <td>26</td>
  </tr>
</table>
```

Default browser display:

Output in the browser

| Menu item | Calories | Fat (g) |
|---|---|---|
| Chicken noodle soup | 120 | 2 |
| Caesar salad | 400 | 26 |

 Use often

# The table Element

`<table> </table>`

- Identifies tabular material
- Contains some number of row (`tr`) elements
- Optionally, may also have a `caption` element and row and column group elements

 Use often

# Table Row (tr) Element

```
<tr> </tr>
```

- `tr` stands for “table row.”
- The only thing that can go between `tr` tags is some number of `th` (header) and `td` (data cell) elements.
- There may be no text content in a row that is not contained within a header or data cell.

Use often

# Table Cells

```
<td> </td>
```

- `td` stands for “table data.”
- Cells can contain any type of web content.
- All content in a table must be contained in **`td`** tags.

 Use often

# Table Headers

```
<th> </th>
```

- `th` stands for “table header.”
- Headers provide information about the cells in the row or column they precede.
- They are key tools for making table content accessible. Screen readers may read headers aloud before each data cell, providing context that is missing when you can’t see the table.
- Headers are often rendered in a bold font by default.

# Table Columns Are Implied

- This table would have 2 rows and 3 columns (because there are 3 cells defined in each row):

```
<table border=1>
    <tr>
      <th>Burgers</th>
      <td>Organic Grass-fed Beef</td>
      <td>Black Bean Veggie</td>
    </tr>
    <tr>
      <th>Fries</th>
      <td>Hand-cut Idaho potato</td>
      <td>Seasoned sweet potato</td>
    </tr>
  </table>
```

Output in the browser

| | | |
|---|---|---|
| **Burgers** | Organic Grass-fed Beef | Black Bean Veggie |
| **Fries** | Hand-cut Idaho potato | Seasoned sweet potato |

*I Add `border=1` just to make the table border visible in this example, in fact you should decorate (style) it in CSS

# Spanning Cells

## Spanning

- Stretching a cell to cover several rows or columns

## Column span

- Stretching a cell to the right over subsequent columns

## Row span

- Stretching a cell downward over several rows

```
<table>
   <tr>
      <th colspan="2">Fat</th>
   </tr>
   <tr>
      <td>Saturated Fat (g)</td>
      <td>Unsaturated Fat (g)</td>
   </tr>
</table>
```

| Fat | |
|---|---|
| Saturated Fat (g) | Unsaturated Fat (g) |

Output in the browser
(* actually without border)

```
<table>
   <tr>
      <th rowspan="3">Serving Size</th>
      <td>Small (8oz.)</td>
   </tr>
   <tr>
      <td>Medium (16oz.)</td>
   </tr>
   <tr>
      <td>Large (24oz.)</td>
   </tr>
</table>
```

| Serving Size | Small (8oz.) |
|---|---|
| | Medium (16oz.) |
| | Large (24oz.) |

# Table Caption

`<caption> </caption>`

- Provides a title or description for the table
- Improves table accessibility
- The `caption` element must appear first in the table element.
- The caption displays above the table by default.

Output in the browser
(* actually without border)

Nutritional Information

| Menu item | Calories | Fat (g) |
|---|---|---|
| Chicken noodle soup | 120 | 2 |
| Caesar salad | 400 | 26 |

```
<table>
   <caption>Nutritional Information</caption>
   <tr>
      <th>Menu item</th>
      <th>Calories</th>
      <th>Fat (g)</th>
   </tr>
   …table continues…
</table>
```

# Row and Column Groups

- For complicated tables, you can create **semantic groups of rows and/or columns** that describe the table's structure.
- **Row group** and **column group** elements also provide more "hooks" for scripting and styling.

# Row Groups

`<thead> </thead>`

`<tbody> </tbody>`

`<tfoot> </tfoot>`

- Provide additional semantic structure
- Row group elements may contain one or more **`tr`** elements (no direct text content).
- Some browsing agents may repeat the header and footer rows on tables that span multiple pages.

*Bidi control codes injected by 'unicode-bidi' at the start/end of 'display: inline' boxes*

| 'unicode-bidi' value | 'direction' value: 'ltr' start | 'ltr' end | 'rtl' start | 'rtl' end |
|---|---|---|---|---|
| 'normal' | – | – | – | – |
| 'embed' | LRE (U+202A) | PDF (U+202C) | RLE (U+202B) | PDF (U+202C) |
| 'isolate' | LRI (U+2066) | PDI (U+2069) | RLI (U+2067) | PDI (U+2069) |
| 'bidi-override'* | LRO (U+202D) | PDF (U+202C) | RLO (U+202E) | PDF (U+202C) |
| 'isolate-override'* | FSI,LRO (U+2068,U+202D) | PDF,PDI (U+202C,U+2069) | FSI,RLO (U+2068,U+202E) | PDF,PDI (U+202C,U+2069) |
| 'plaintext' | FSI (U+2068) | PDI (U+2069) | FSI (U+2068) | PDI (U+2069) |

* The LRO/RLO+PDF pairs are also applied to the root inline box of a block container if these values of 'unicode-bidi' were specified on the block container.

```
<table>
…
<thead>
  <!-- headers -->
  <tr></tr>
  <tr></tr>
  <tr></tr>
<thead>
<tbody>
  <!-- data -->
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
</tbody>
<tfoot>
  <!-- footnote -->
  <tr></tr>
</tfoot>
</table>
```

# Column Groups

`<colgroup> </colgroup>`

`<col> </col>`

- Allows you to assign `id` and `class` names to columns so they can be accessed by scripts or styles
- `colgroup` elements go at the start of the table, after the `caption` element if there is one.
- `colgroup` elements contain no content; they only provide an indication of column structure

The number of columns in a group is noted with the **`span`** attribute:

```
<table>
  <caption>…</caption>
  <colgroup></colgroup>
  <colgroup span="2"></colgroup>
  <colgroup span="2"></colgroup>

  <!-- rest of table... -->
```

If you need to access individual columns, identify them with **`col`** elements:

```
<colgroup></colgroup>
  <colgroup>
    <col class="start">
    <col class="end">
  </colgroup>
  <colgroup>
    <col class="start">
    <col class="end">
  </colgroup>
```

# End of the topic

HTML for structure