Semantic HTML and Page Organization

THRIVE
Women for Women

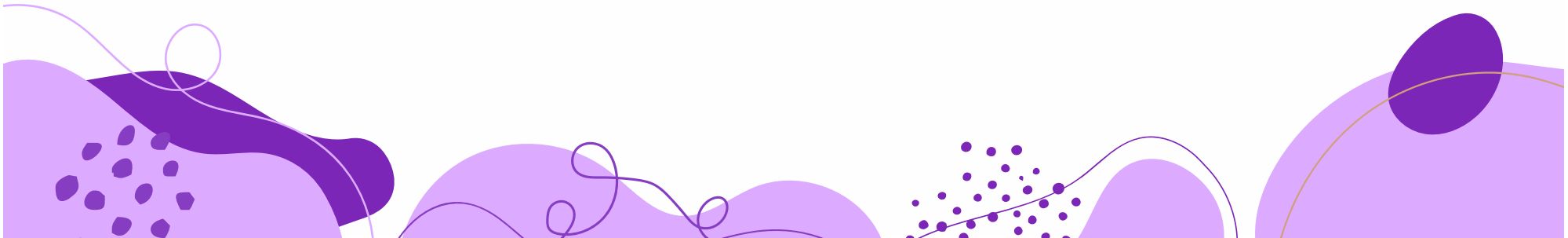
Session Objectives

- Recap key concepts from Week 2
- Learn to use semantic HTML elements
- Embed media
- Apply basic accessibility principles
- Paths & Linking
- Organize clean HTML code
- Practice with hands-on exercises

Recap of Week 2

Week 2 - HTML Basics

- HTML document structure
- Tags: headings, paragraphs, links, lists



What is Semantic HTML?

Semantic HTML

- They clearly describe their purpose/meaning in the context of the content.
- They improve accessibility and SEO.
- They are easier to read and maintain, since the structure of the page is evident from the tags.

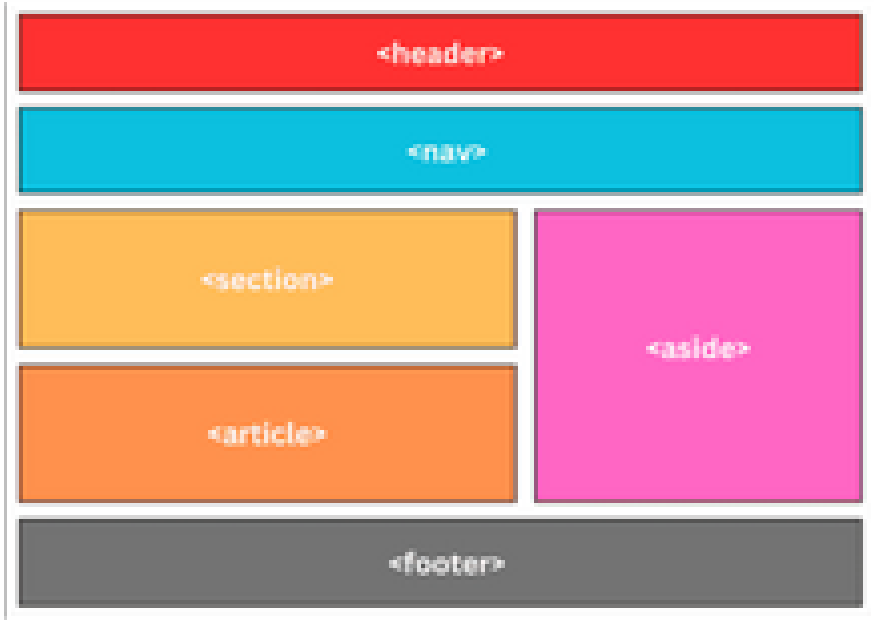
Semantic HTML

```
<!-- Non-semantic: using divs with classes/IDs -->
<div id="header"> ... </div>
<div id="nav"> ... </div>
<div id="main-content"> ... </div>
<div id="footer"> ... </div>

<!-- Semantic HTML5 elements for the same structure: -->
<header> ... </header>
<nav> ... </nav>
<main> ... </main>
<footer> ... </footer>
```

Common Semantic HTML Elements

- <*header*>
- <*nav*>
- <*main*>
- <*section*>
- <*article*>
- <*aside*>
- <*footer*>



Common Semantic HTML Elements

- <*header*> : usually contains introductory content like site name. we can have it for the page and also inside individual articles or sections as needed.
- <*nav*> : typically, this is the site menu for the page. It usually contains a list of links to important sections of the site.
- <*main*> : contains the central content for your page.
- <*section*> : section of related content, typically with a heading. we could use it to break the <main> content into logical blocks.

Common Semantic HTML Elements

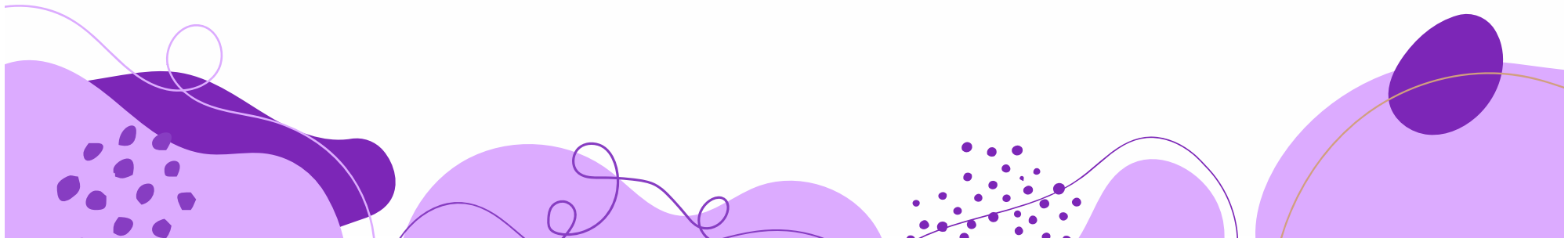
- <*article*> : independent, self-contained piece of content. It should make sense on it's own. It could be used for things like blog entries in a news site.
- <*aside*> : content aside form the main content. It's not part of the primary narrative, but extra/ complimentary information.
- <*footer*> : typically contains the closing or fine-print content. e.g. copyright info, short site map.

Common Semantic HTML Elements

Semantic vs Non-Semantic Recap

- Non-semantic elements like <*div*> and <*span*> tell nothing about their content by themselves.
- Semantic elements clearly define their content and role (header, footer, etc.), which improves code clarity and accessibility.
- Whenever possible, prefer a semantic element like <*nav*> over a plain <*div* class="menu">.

  This is considered a best practice in modern web development.



Embedding Media: Images, Audio, Video, and Iframes

Embedding Media: Images, Audio, Video, and Iframes

Beyond text, real webpages include media like images, sounds, vidoes and contents from other sites.

- **Images** <*img*> : is used to embed images. it's a self closing tag that requires at least a src attribute and alt attribute.

```
<img src="images/team.jpg" alt="Group photo of the team members" />
```

Embedding Media: Images, Audio, Video, and Iframes

HTML5 introduced the <audio> and <video> elements for embedding media without needing extra plugins. These tags allow you to embed media files (like .mp3, .mp4, etc.) with built-in browser controls.

Audio <audio> : is used to add sound. We use <audio> with a src or nested <source> tags, and include the controls attribute so the user gets playback controls (play/pause, scrub, volume).

– The controls attribute is what displays the play button and timeline.

Embedding Media: Images, Audio, Video, and Iframes

Video <video> : is used to embed video. Common attributes are controls, width/height, and optionally poster (an image to show before the video loads).

```
<audio controls>
  <source src="song.mp3" type="audio/mpeg">
  <source src="song.ogg" type="audio/ogg">
  Your browser does not support the audio element.
</audio>

<video src="promo.mp4" width="640" height="360" controls poster="promo-thumb.jpg">
  Sorry, your browser doesn't support embedded videos.
</video>
```

Embedding Media: Images, Audio, Video, and Iframes

Iframe <iframe> : is like a mini browser window embedded within your page.
- It lets you display another HTML page or resource inside a box on your page.
- Common use-cases include embedding a Google Map, a YouTube video (via YouTube's embed code), or even an external webpage.

```
<iframe width="560" height="315"
  src="https://www.youtube.com/embed/VIDEO_ID"
  title="YouTube video player" frameborder="0"
  allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
  allowfullscreen>
</iframe>
```

Accessibility Basics in HTML

Accessibility Basics in HTML

Building web pages with accessibility in mind ensures that people with disabilities can use your site. Here are some accessibility best practices related to HTML:

Alt Text for Images:
– Describes the content in a concise phrase.
– This is read by screen readers to users who can’t see.
– Never leave out the alt attribute, or screen readers may announce the file name which is often useless (e.g., “IMG_1234.jpg”).

Accessibility Basics in HTML

Form Labels: When you include forms like text inputs, each input should have a corresponding <*label*> that clearly describes what the input is for.
– The label can be tied to the input by using the for attribute (matching the input's id).

```
<label for="email">Email Address:</label>
<input type="email" id="email" name="email">
```

Accessibility Basics in HTML

Structured Headings: Use headings <h1> ... <h6> in a logical, nested order to represent the page outline.
– There should usually be one <h1> (the main title of the page), then <h2> for major sections, and so on.
– Don’t choose heading levels based on appearance.
– A proper hierarchy helps screen reader users navigate the page by headings (they often jump through headings) and also benefits SEO.

Accessibility Basics in HTML

Landmark Roles (Implicit): By using semantic elements like <header>, <nav>, <main>, <aside>, <footer>, you are implicitly creating landmarks that assistive technologies can use to skim the page.

Link Text: Ensure your hyperlink text makes sense out of context.
– Avoid using vague link text like “Click here” or “Read more” by itself. Instead, make the clickable text descriptive.
– e.g., Visit our About Us page instead of Click here for About Us.

Using Relative vs. Absolute Paths

Relative vs. Absolute Paths

Absolute Path (URL): An absolute path is a full URL that includes the protocol (http/https) and domain name.
– It points to a resource on the web no matter who is accessing it or from where.
– Use absolute URLs for linking to external websites or resources hosted on another site.

```
<img src="https://www.example.com/images/logo.png" alt="ExampleCorp Logo">
<a href="https://twitter.com/ThriveClub">Our Twitter</a>
```

Relative vs. Absolute Paths

Relative Path: A relative path points to a file relative to the current HTML file's location.
– It's like giving directions based on where you currently are.
– For example, if your website folder has an images subfolder and inside it a file photo.jpg, you could reference it as:

```
<img src="images/photo.jpg" alt="Sample Photo">
```

Relative vs. Absolute Paths

– Use relative paths for any internal links within your project such as images, other HTML pages, or CSS/JS files.
– The paths make local testing easier since you're not relying on fixed web URLs.
– In contrast, use absolute paths when linking to external resources, like websites or embedded content from another domain (e.g., iframes or external scripts).

Writing Clean, Organized HTML Code

Clean, Organized HTML Code

Indentation and Whitespace: Use consistent indentation to show nesting of elements.
– Typically, 2 spaces or 4 spaces per indent level is standard (pick one and stick to it).
– Do not mix tabs and spaces. In many teams, spaces are preferred (and our style will follow that). Indent child elements inside parent elements.

Consistent Naming & Structure: Name your files and IDs/classes in a consistent way.

Clean, Organized HTML Code

HTML Comments: Use comments (<!-- comment text -->) to annotate the code, especially to indicate structure or important notes.
– Another use is temporarily disabling a piece of code by commenting it out.
– Comments do not display on the page; they are just in the code for humans.

Avoid Inline Styling (for now): Keeping structure (HTML) separate from presentation (CSS) leads to cleaner code.

Clean, Organized HTML Code

Validate and Tidy Up:

– There are HTML validator tools (like the W3C Markup Validator) that can check your code for errors.

– Using these can catch un-closed tags or nested mistakes.

– Also, formatters (including VS Code auto-format on save) can automatically indent/align your code. But it’s best to practice doing it manually to build good habits.

Q/A