

HTML 101

Getting Started with HTML

What is HTML?

A standard webpage comprises three distinct layers that collaborate to provide a user experience. These layers consist of:

- The content layer, presenting the visible information on the page.
- The presentation layer, which manages the visual appearance of the information.
- The behavior layer, facilitating user interaction with the page.

HyperText Markup Language (HTML) forms the content layer and serves as the fundamental structural foundation of a webpage. It is a universal language employed across all websites. Understanding HTML is crucial whether you aim to create your website, web application, or even modify existing websites and applications.

Cascading Style Sheets (CSS) govern the presentation layer, determining the visual style elements such as colors, typography, and layout. JavaScript, on the other hand, handles the behavior layer by introducing interactivity, such as enabling larger image displays upon user interaction, like clicking a small image.

In this learning module, the basics of HTML is discussed in depth. To start, let's break down the acronym HTML: Hypertext Markup Language. Hypertext refers to text with embedded links connecting it to other documents or texts, forming the interconnected web we navigate daily.

HTML is a markup language instrumental in instructing browsers on how to display text, links, images, and videos, serving as the fundamental building block for all web-related content and applications.

To grasp the significance of HTML, consider how word processing software like Microsoft Word or Google Docs enables formatting text, creating headings, adjusting text size, inserting links, and organizing content. Similarly, HTML uses markup tags to structure and format content for web browsers. Without these tags, web content lacks structure and coherence.

Inspecting the HTML source code of a webpage reveals the markup tags surrounding the content. Despite being instructions for the browser, these tags are intuitively named for human comprehension, such as title, body, header, main, and footer. HTML's universality across different browsers, from desktops to mobile devices, underscores its pivotal role in web development.

By the course's end, you'll acquire the skills to craft a simple website's HTML structure. Remember, while HTML lays the groundwork, incorporating CSS can transform a page's structure into an aesthetically pleasing design.

HTML Structure and Composition

This section outlines the standard structure of web pages using HTML, emphasizing the essential components required to create a basic webpage. It highlights the necessary steps to start coding, such as:

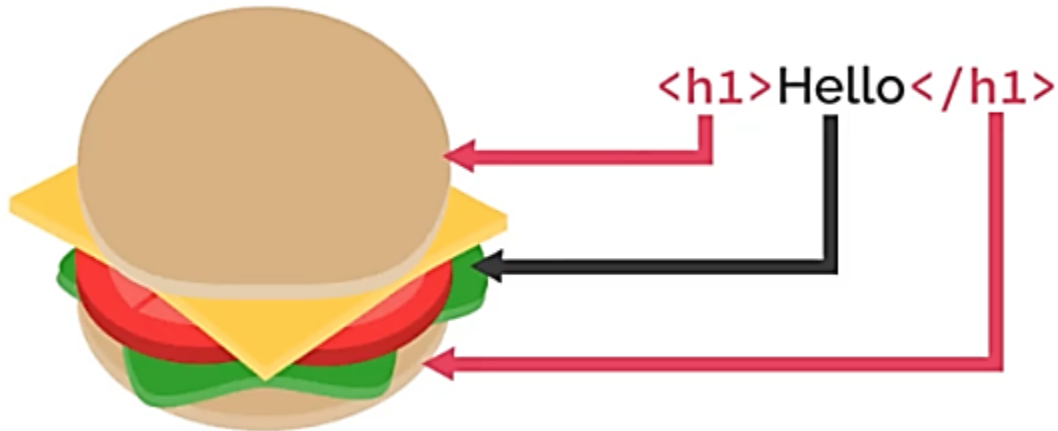
1. Introduction to the common structure of web pages, involving instructions for browsers, general page information, and content placement (images, text, links).
2. The creation of an HTML file named `index.html`, signifying the document type and the root elements used in HTML.
3. An explanation of the HTML structure, including the head element for page information (like the title) and the body element for visible content.
4. Practical demonstrations of adding content within the head and body elements using tags like title and text.
5. Emphasis on the importance of HTML tags to instruct browsers on displaying content, demonstrating how tags structure text and content for better visibility.
6. Guidance on referencing HTML elements using resources like MDN (Mozilla Developer Network) HTML reference for exploring available elements and their functions.
7. Encouragement to explore heading and paragraph elements in MDN as a precursor to upcoming lessons.
8. Reminder to maintain the unique occurrence of HTML, head, title, and body elements per webpage.
9. Helpful tips on creating HTML templates to streamline coding.

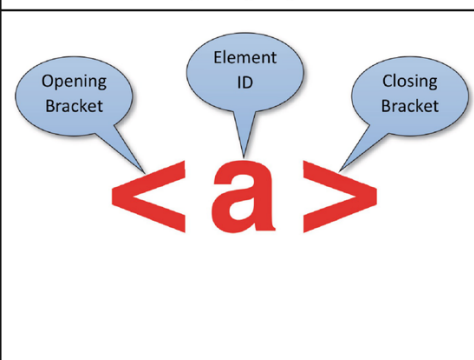
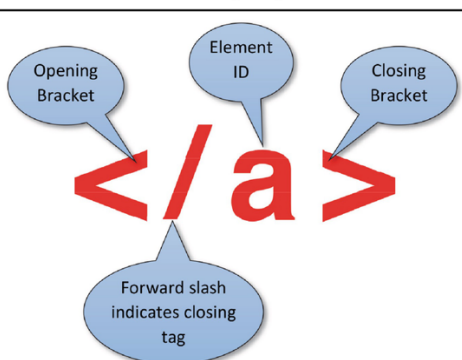
The tutorial focuses on establishing a foundational understanding of HTML's structure and its crucial role in web development, aiming to facilitate the learning process for creating simple webpages.

HTML structure

An HTML **tag** is a markup instruction identified by <, the tag name, and >.

- An **opening tag** indicates the element's starting point in the document.
- A **closing tag** indicates the element's ending point in the document.



| Opening Tag | Closing Tag |
|---|--|
|  |  |

HTML document

The HTML Living Standard, produced by WHATWG, defines the minimal parts of an HTML document:

1. The **<!DOCTYPE html>** declaration instructs the web browser about what type of document follows.
2. The **<html>** element encloses everything but the `<!DOCTYPE html>` declaration. `<html lang="en">` indicates that the document's language is English.

3. The **<head>** element contains the document title, document metadata, and various other elements that are typically not displayed in the webpage.
4. The **<meta>** element specifies metadata, which is data that describes the document's data. `<meta charset="UTF-8">` describes how characters are represented in the HTML document. Additional `<meta>` tags may be used to indicate when the document was saved, who the author is, etc.
5. The **<title>** element specifies the document's name. The title is usually displayed in the browser's titlebar, is used by search engines, and is used for bookmarking.
6. The **<body>** element encloses all elements and content to be rendered in the browser.

The opening `<html>` tag uses an attribute to indicate the webpage's language, and `<meta>` uses an attribute to indicate the character set. An element **attribute** provides additional information about the element and is included only in the opening tag. An attribute has a name and a value, specified using the form: `name="value"`. Ex: `<meta charset="UTF-8">` has an attribute named `charset` with value "UTF-8".

```
<!DOCTYPE html>
```

```
<html>
```

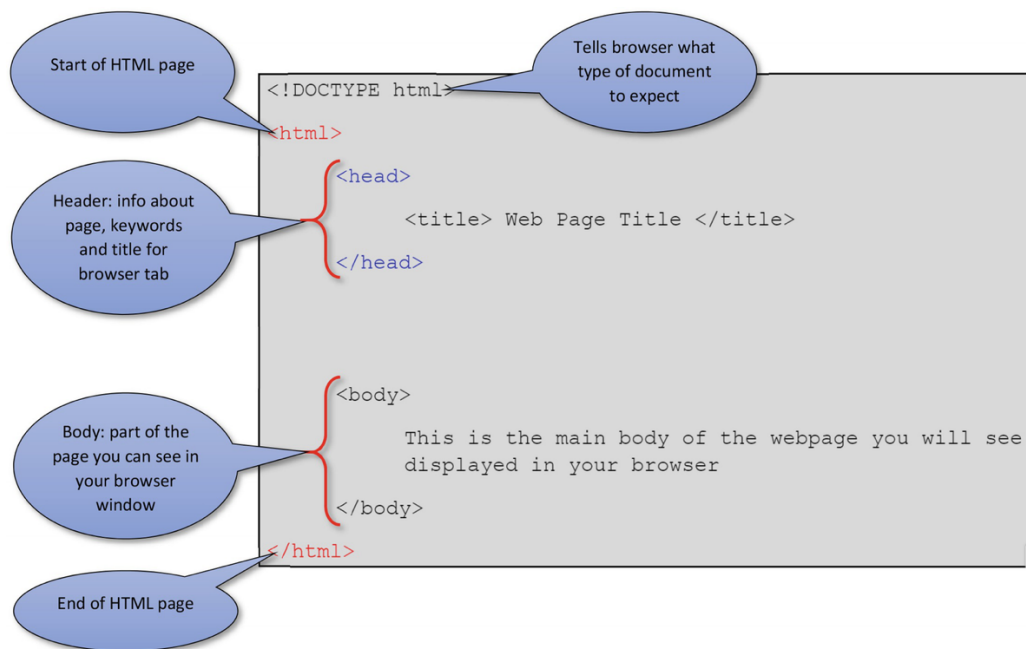
```
<head>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```



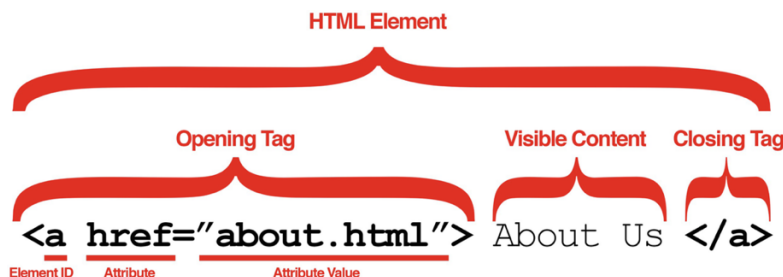
Paragraphs

The `<p>` element in HTML serves to create paragraphs within a document, delineating blocks of text with appropriate spacing for readability. Each paragraph is encapsulated by an opening `<p>` tag and a closing `</p>` tag, ensuring proper rendering by web browsers.

However, it's essential to avoid common errors like forgetting the closing `</p>` tag or attempting to nest one paragraph within another. While browsers might visually accommodate nested `<p>` tags, this practice leads to invalid HTML, flagged by validators. Maintaining proper structure not only ensures consistency across browsers but also promotes accessibility and adherence to web standards.

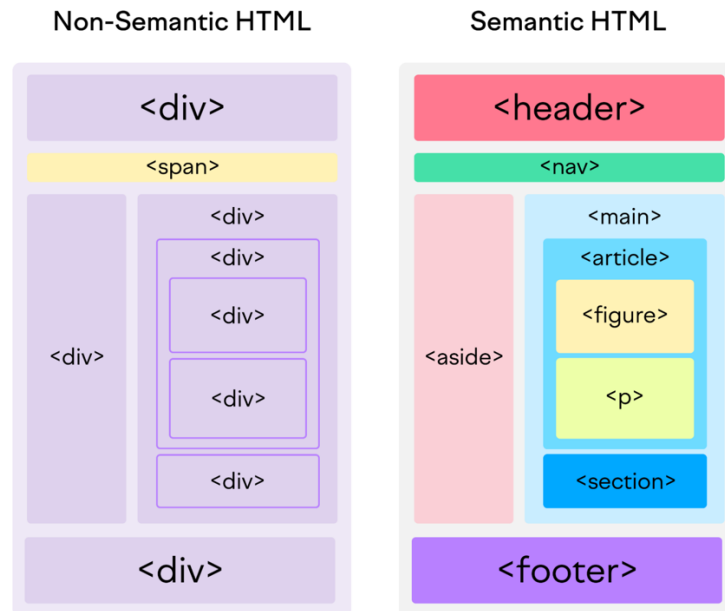
```
<p>First Paragraph</p>
<p>Second Paragraph</p>
<p>Third Paragraph</p>
```

Attributes



Source: The Absolute Beginner Guide to HTML and CSS, Kevin Wilson, Oreilly.

What Is Semantic HTML?



Headings and Paragraphs

The classroom section explains how HTML, through markup tags, structures content for browsers, enhancing readability. It illustrates the importance of structuring content for a better reading experience, likening it to how books and articles use titles, subheadings, and paragraphs. Key points covered include:

Introduction to HTML's role in organizing content through tags.

Demonstrations on structuring content using tags like headings (e.g., **h1** for main headings) and paragraphs (**p** tags).

Emphasis on **h1** as the highest level heading, making text noticeable, and **h2** as a slightly less significant heading, displayed slightly smaller.

Explanation of various heading levels (**h1** through **h6**) representing different content importance levels.

Overall, the tutorial focuses on structuring content effectively using HTML tags like headings and paragraphs, providing a foundational understanding of how markup improves content organization on web pages.

See `html-1.html` file

Exercise:

For the following HTML code:

```
<!DOCTYPE html>
<html>

<head>
  <title>Headings and Paragraphs</title>
</head>

<body>
  This is the Main Headline!
  Oat cake chocolate bar jelly. Tootsie roll cheesecake sweet gummies candy cookie
  pudding cotton candy carrot cake.
  Souffl&eacute;; caramels brownie oat cake cheesecake.

  Level 2 Heading
  Ice cream candy canes muffin icing pudding muffin jelly topping carrot cake. I
  love gingerbread dessert jujubes
  bonbon cupcake tootsie roll I love. Oat cake topping caramels I love cupcake oat
  cake chocolate topping donut.

  Level 3 Heading
  Cotton candy topping halvah sugar plum gummies souffl&eacute;;. Ice cream danish
  donut sugar plum. Macaroon carrot
  cake gummies. Caramels oat cake chocolate cake.
</body>

</html>
```

- 1- Set the main headline to a heading level 1 element. Then, place the line of text below the main headline inside opening and closing paragraph tags.
- 2- Next, set the subheading to a heading level 2 element and the line of text below it to a paragraph element.
- 3- Finally, set the third heading to a heading level 3 element and the text below it to a paragraph.

Text formatting

A number of HTML elements cause the enclosed text to render in a different font in the browser:

The `` element indicates emphasized text, such as text having an emphasized pronunciation when spoken, and is italicized by default.

The `` element indicates text that has strong importance, and is bolded by default.

The `<cite>` element denotes a title, such as a book or song title, and is italicized by default. Ex: `<cite>Spaceballs</cite>` is a parody of the `<cite>Star Wars</cite>` trilogy. yields: Spaceballs is a parody of the Star Wars trilogy.

The `<mark>` element denotes important content that should be semantically highlighted and is rendered with a yellow background by default. Ex: `<mark>Highlight</mark>` what is important. yields: Highlight what is important.

The `` element indicates text that needs attention, like key words in a document abstract or product names in a review, and renders the text in bold. Ex: Mix the flour and oil together.

The `<i>` element indicates text in an alternative voice, such as a word or phrase in a foreign language, and is rendered using italics. Ex: Dashi is a stock used in Japanese cooking.

The `<u>` element denotes text that should appear differently from normal text, such as misspelled words, and is underlined by default. Ex: Misspelled is often misspelled as misspelled.

| Element | HTML example | Rendered | Semantics |
|---------|--|------------------|--|
| em | <code>emphasis</code> | <i>emphasis</i> | Emphasized text |
| cite | <code><cite>cite</cite></code> | <i>cite</i> | Title of a work |
| strong | <code>strong</code> | strong | Important text |
| mark | <code><mark>mark</mark></code> | mark | Marked or highlighted text |
| var | <code><var>variable</var></code> | <i>variable</i> | Definition of a variable in a computer program |
| kbd | <code><kbd>keyboard</kbd></code> | keyboard | Keyboard input |
| code | <code><code>code</code></code> | code | Computer code |
| samp | <code><samp>sample</samp></code> | sample | Sample output from a computer |
| b | <code>bold</code> | bold | Bold text |
| i | <code><i>italic</i></code> | <i>italic</i> | Text of an alternate voice or word from another language |
| u | <code><u>underline</u></code> | <u>underline</u> | Text that is rendered differently from normal text |

Creating Lists

This section introduces **lists** in HTML, emphasizing their significance in web design and development. It covers:

- Explanation of HTML's three list elements: **** for unordered lists, **** for ordered lists, and **<dl>** for description lists (though not covered extensively in this course).
- Demonstration of creating a website section list (Home, About, Articles, Contact) using the **** element and marking up individual items with **** tags for unordered lists.
- Clarification on how the **** element displays items in an unordered manner with bullet points.
- Introduction to the **** element for ordered lists, showing how it displays items with numbers in sequential order.
- Explanation of nesting lists (unordered inside ordered lists), showcasing nesting with the **** tag within the Contact list item to create a multi-level list.
- Recap on the characteristics of unordered and ordered lists, their respective display styles (bullets and numbers), and the option to nest lists to create structured outlines.

Lists play a pivotal role in web design and front-end web development. Virtually every website or application leverages lists to showcase navigation menus, items in shopping carts, movie catalogues, and various other content.

HTML offers three key elements for crafting lists: **** for unordered lists, **** for ordered lists, and **<dl>** for description lists. While the **<dl>** element aids in organizing terms within descriptions (similar to a glossary), it's less commonly used compared to the other list types.

To create a list, include a set of opening and closing **** tags, and within the ****, input:

- Home,
- About,
- Articles, and
- Contact.

When considering a list, the expectation is for each item to be placed on its individual line. However, the **** tag, on its own, doesn't exhibit a standard list format. As observed in the browser, all items appear on a single line. To delineate the individual items within a list, enclose them within **** tags, representing list items. Returning to our list, let's envelop each menu item with opening and closing **** tags. Each **** tag now signifies an item in the list, and when **** tags are contained within a **** tag, they are displayed using bullet points.

An unordered list can be placed within another list to generate multi-level lists. After expanding the contact list item, just beneath the word 'contact,' let's create a nested unordered list by incorporating a set of opening and closing `` tags. Within this list, generate three list items, but feel free to include as many as needed.

Following that, generate a list item for 'Phone' and another list item for 'Address.' After saving the file, observe in the browser how nesting a list results in the browser implementing indentation to enhance the list's structural organization. Consequently, you'll notice how the list now mirrors an outline format.

See `html-2.html` file

Creating Links

No site is truly complete without hyperlinks. Hyperlinks are the web's core connecting elements, enabling you to leap from one page to another or from one site to another, in pursuit of information. Imagine the web without hyperlinks; it would be just a vast array of isolated pages.

To transform any text into a hyperlink, you use the `'a'` tags. For instance, beneath the paragraph above our `h2`, we insert a pair of `'a'` tags with the text "Start Your VR Journey." The `'a'` tag creates an anchor element; this is why it's named `'a'`. It's utilized to link a URL to a specific text piece. Consequently, everything between the opening and closing anchor tags becomes a clickable link, directing users to different webpages or locations on the current page.

Currently, in the browser, these changes are not yet visible, and clicking the text does nothing. To make the browser recognize it as a link, we need to provide additional information, such as the web address it should link to. HTML elements accept attributes that enhance their meaning and behavior. Attributes inform the browser about details like where the text enclosed in `'a'` tags should link to, or whether to open the link in a new tab or window.

Let's begin with the `'href'` (hypertext reference) attribute, which is specific to anchor elements. It points to a location and directs the user there by changing the URL in the address bar. Attributes are written inside an element's opening tag. So, after the `'a'` in the opening `'a'` tag, we add `'href'`. An attribute is always followed by an equals sign and a value, enclosed in single or double quotation marks.

For the `'href'` value, we specify the destination or URL of the link. For this example, we'll link to the VR page of Treehouse. Once the `href` attribute is added, the browser recognizes the text as a hyperlink and displays it accordingly, usually in blue and underlined.

Hyperlinks can point to various locations, like another website, a different page on the same site, a different location within the same page, open in a new window or tab, or even trigger an email composition in the user's default email program.

When linking to a different website, we use an 'absolute URL' as the href value. Absolute URLs provide the browser with all the necessary information to navigate the user to a web location. This format typically begins with '**http://**' or '**https://**', followed by the domain name and the path to the specific webpage.

By default, clicking a link navigates to the new location in the current window or tab. However, to open links in a new tab or window, keeping users on your site even after they've visited the linked site, we use the 'target' attribute with the value '**_blank**'. Adding this to our 'a' tag, the link now opens in a new tab.

Sometimes, you might see links with just a hash or pound symbol as the **href** value. This symbol can link users to different parts of same page or serve as a placeholder while building the page.

Next, try creating links for the top resource items on your site, directing them to any URL of your choice. You might also want to experiment by turning items like 'home', '**about**', '**articles**', and 'contact' into placeholder links using the hash symbol. In the upcoming stage, we'll transform this list into our main navigation menu. You can find examples of how I added these links in the teacher's notes of this video.

In this stage, you've built a basic webpage that will serve as the foundation for the VR website. Up next, we'll delve into HTML elements that divide a page into logical sections and group related elements. You'll learn about the significance of planning your HTML structure and using markup that accurately represents your content's semantics.

See html-3.html file

Exercise

For the HTML code:

```
<!DOCTYPE html>
<html>

<head>
  <title>Lists and Links</title>
</head>

<body>

  Appetisers
  Foods
  Desserts

</body>

</html>
```

- 1- Convert the text inside the <body> tags into an unordered list.
- 2- Make the text inside each list item a link. The first item should link to ***example.com/appetisers***, the second to ***example.com/foods*** and the third to ***example.com/desserts***.

Structuring Content

Semantic HTML: <header>, <footer> and <section>

In web design and development, HTML serves to structure and give meaning to content. For instance, a '**p**' tag signifies a paragraph, while an '**h1**' tag indicates the primary heading of the page. Writing markup that conveys the essence of your content is known as writing semantic markup. The concept of semantic markup or HTML is frequently discussed in web design and front-end development.

Semantics in HTML refers to the notion that markup should describe the content's meaning rather than its presentation or appearance. For example, using an '**h1**' tag merely to make text appear large and bold, even though it's not a main heading, is inappropriate. '**h1**' and '**p**' tags have distinct meanings. Semantic markup enhances your website's clarity for search engines, helping them categorize your pages and aiding users in finding your content.

Moreover, semantic markup improves accessibility for users who depend on assistive technologies. Users with visual impairments, for instance, rely on screen readers that audibly read the page's content. Properly written semantic markup aids these users in navigating and interacting with your site.

As developers, it's crucial to be mindful about our markup, ensuring that each tag is apt for its content. This course will delve further into semantic markup. We'll begin with HTML's set of elements that define content sections.

A typical webpage has three main sections:

- 1- a *header*,
- 2- a *footer*, and
- 3- the *main* content area.

The header, usually at the page's top, contains the site logo, navigation, main heading, and site description. The footer, at the bottom, includes information like the company address, site authors, copyright date, and additional navigation. The main content, located between the header and footer, holds the most vital information, such as blog entries, videos, articles, photo galleries, recipes, etc. This is the content that draws users to the page.

Often, the header and footer content remains consistent across pages, while the main content area varies. Elements like **'body'**, **'head'**, and **'title'** use straightforward, human-readable words to describe their content. HTML's semantic sectioning and content grouping elements, like **'header'**, **'footer'**, **'section'**, and **'article'**, serve similar purposes, indicating their role in the document.

In the `index.html` file, we'll logically divide our page into sections, starting with the header. Place the introductory content inside a `'header'` tag, located above the **'h1'** element. Similarly, create a **'footer'** at the page's bottom, containing a paragraph with the blog's name and copyright date, using an HTML character entity for the copyright symbol.

The main content area, between the header and footer, often divides further into subsections. For instance, group blog entries in a **'section'** element. The first element in each section should be a heading that describes its topic. Add more sections as needed, such as for **'About'** and **'Contact'** content, each defined by **'section'** tags and accompanied by relevant headings and text.

There's flexibility in using sections; they're meant for organizing content logically. Sections also aid assistive technologies in identifying distinct parts of a page. Other specific sectioning elements, like the **'article'** tag, cater to more specific content grouping, which will be covered in upcoming videos.

See semantic-1.html

Sectioning Content with `<article>`, `<nav>` and `<aside>`

Let's continue structuring our content logically using HTML's content sectioning elements. This overview will cover how to effectively use the **article**, **nav**, and **aside** tags.

The **article** tag is ideal for self-contained content pieces, like individual blog entries in our VR-themed section. Each blog entry, being an independent article, should be enclosed in **article** tags. This markup is akin to an article in a magazine or newspaper, understandable and complete on its own, even when separated from its original context. The **article** element is perfect for content that could be syndicated, shared on social media, or featured on sites like Reddit. This includes blog posts, magazine articles, forum posts, and similar standalone content pieces. Importantly, each **article** should start with a heading for clarity.

The **nav** tag is specialized for sections with navigation links, such as a list of links to other pages or parts of the same page. We currently use an unordered list (**ul**) for site navigation; this should be wrapped in **nav** tags. Remember, **nav** should only be used for main navigation sections, not every link. It's acceptable to have multiple **nav** elements for different navigation purposes, like in the header and footer. The **nav** element is especially helpful for users with assistive technologies, aiding them in finding navigation sections quickly.

Lastly, the **aside** tag is for content tangentially related to the main topic but not central to it. This content, often found in sidebars, includes things like Twitter feeds or lists of related articles. On our page, we have a list of top resources above the contact section, which, while related to the overall theme, isn't central to the main content. We'll encapsulate this in **aside** tags. Use **aside** when you have content that supports but isn't the primary focus of the page.

See semantic-2.html

Exercise:

For the code:

```
<!DOCTYPE html>
<html>

<head>
  <link href="styles.css" rel="stylesheet">
  <title>John Doe's Creative Portfolio</title>
</head>

<body>
  <ul>
    <li><a href="#">Profile</a></li>
    <li><a href="#">Projects</a></li>
    <li><a href="#">Get in Touch</a></li>
  </ul>
  <h1>Welcome to My Design & Development Portfolio!</h1>
  <p>Explore my world of innovative and engaging web solutions.</p>

  <h2>Introduction</h2>
  <p>Vivamus et justo nec nulla facilisis ullamcorper. Curabitur ut turpis a enim
convallis volutpat. Integer euismod
    magna a risus lacinia, a pharetra magna interdum.</p>
  <ul>
    <li><a href="#">Featured project #1</a></li>
    <li><a href="#">Featured project #2</a></li>
    <li><a href="#">Featured project #3</a></li>
  </ul>

  <p>&copy; 2023 John Doe's Portfolio</p>
  <p>Connect with me on <a href="#">LinkedIn</a>, <a href="#">Facebook</a>, and <a
href="#">Behance</a></p>
</body>

</html>
```

- 1- Place the **ul**, **h1** and **p** elements at the top of the page inside an element that represents a group of introductory content.
- 2- Place the paragraphs at the bottom of the page inside an element that typically contains information about the site, copyright data or related links.
- 3- Place the content between the header and footer (**h2**, **p** and **ul**) inside an element that represents standalone sections of content.
- 4- Finally, place the top **** inside an element that represents a major section of navigation.

Marking Up a Blog Post

Let's build a new blog post page for our website, applying the HTML tags we've learned to structure the content semantically. We'll start by creating a new HTML file named **article.html**. To speed up the process, we can copy the HTML from **semantic-2.html** and modify it accordingly.

In **article.html**, we'll first update the title tag to "VR Article." We'll retain the header and footer from the copied content but remove everything in between them. This new page will focus on a standalone article that could potentially be syndicated to other sites.

To structure our article, we'll use the **article** element, indicating that the content is a self-contained piece. We'll start with an **h2** tag titled "VR Article" followed by paragraphs of placeholder text, which can be sourced from the teacher's notes.

We'll also explore the use of the **aside** element. When placed inside an **article**, **aside** holds content related to but not central to the article, like a pull quote. We'll include a pull quote using the **q** tag, which automatically adds quotation marks.

Alternatively, when **aside** is used outside of an **article**, it pertains to content related to the entire page, such as a sidebar with links to related articles. Below our **article** tag, we'll add an **aside** containing an **h3** tag and a list of links to additional VR articles. This list, although important, isn't a primary navigation component, so we won't wrap it in a **nav** element. Finally, to link the "Read more" links on the homepage to our new article, we'll add **article.html** as the **href** value in the corresponding links in **index.html**.

By following these steps, you've created your first blog post using semantic HTML. This approach not only structures your content logically but also enhances accessibility and SEO. More details on linking web pages will be covered in upcoming lessons.

See semantic-3.html

Special Character

Entities

An **entity** is a mechanism for writing special characters or symbols in HTML, such as mathematical symbols, characters in most languages, and many other symbols. Many HTML entities can be specified by name. However, some entities do not have a name and can instead be specified using a decimal or hexadecimal number as shown in the table below.

| Format type | Format example | Symbol | Description |
|--------------------|----------------------------|--------|--|
| Entity name | <code>&copy;</code> | © | HTML defines names for common well-known entities, which are typically English abbreviations for the symbol. |
| Decimal number | <code>&#169;</code> | © | Every HTML entity has a unique number, which can be specified as a decimal value. |
| Hexadecimal number | <code>&#x000A9;</code> | © | Every HTML entity has a unique hexadecimal number, which is the hexadecimal equivalent to the entity's decimal number. |

Emoji

Emoji characters can be displayed in a webpage with a decimal or hexadecimal entity. Ex: The happy face emoji, 😊, can be displayed with `😀` or `😀`. Since emoji are characters, not images, emoji characters can also be copied and pasted into any HTML document. Ex: `<p>😊</p>`.

Grouping Content with `<main>` and `<div>`

The **main** element in HTML is used to define the primary content of a webpage, assisting screen readers and other assistive technologies in identifying the core content. In our example, the main content includes sections about the site, articles, and contact information. We enclose these sections in a **main** element, placing the opening tag before the 'About' section and the closing tag after the 'Articles' section. The 'Contact' section is also moved inside the **main** element.

For content organization and layout control, we use the **div** element, a generic container with little semantic meaning, ideal for grouping related content. In our example, the introductory paragraph and link below the header don't form a new content section nor are they the main focus of the page, making them suitable for wrapping in a **div** tag. Additionally, a **div** can be used as a wrapper for the entire page content, allowing for easier layout management and styling with CSS.

For quotations, HTML provides two tags: **q** for short inline quotes and **blockquote** for longer, multi-line quotations. If quoting an excerpt from a book or article, the **blockquote** tag is appropriate. In our example, we place a well-known VR quote by Mark Zuckerberg inside

blockquote tags within the **aside** element. The **cite** attribute within the **blockquote** tag can reference the quote's source URL, while the **cite** tag within the **blockquote** can credit the author, in this case, linked to the source.

A **footer** tag can also be used within a **blockquote** to attribute the author or source, demonstrating that **footer** elements aren't limited to the bottom section of a page but can be used multiple times for different purposes.

In summary, we've organized the page's main content using the **main** element, grouped non-semantic content with **div** tags, and correctly attributed quotes using **blockquote**, **cite**, and **footer** elements. This approach enhances the semantic structure of the page, making it more accessible and meaningful for both users and search engines.

See semantic-4.html

Exercise:

For the code below:

```
<!DOCTYPE html>
<html>

<head>
  <link href="styles.css" rel="stylesheet">
  <title>My Blog</title>
</head>

<body>
  <header>
    <h1>My Web Design & Development Blog!</h1>
    <nav>
      <ul>
        <li><a href="#">About</a></li>
        <li><a href="#">Articles</a></li>
        <li><a href="#">Recent Work</a></li>
      </ul>
    </nav>
  </header>

  <h2>The Main Articles</h2>

  <h3>My Favorite HTML Courses</h3>
  <p>Fusce semper id ipsum sed scelerisque. Etiam nec elementum massa.
  Pellentesque tristique ex ac ipsum hendrerit,
    eget <a href="#">feugiat ante faucibus</a>.</p>
```

```

    <h3>10 Handy CSS Features</h3>
    <p>Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac
    turpis egestas. Vestibulum ante
        ipsum primis in faucibus orci luctus et <a href="#">ultrices
    posuere</a>.</p>

    <h3>Follow Me on Social Media:</h3>
    <ul>
        <li><a href="#">Twitter</a></li>
        <li><a href="#">Facebook</a></li>
        <li><a href="#">LinkedIn</a></li>
    </ul>

    <footer>
        <p>&copy; 2017 My Blog</p>
    </footer>
</body>
</html>

```

- 1- Group each set of **h3** and **p** elements using an element that describes self-contained pieces of content, (using **<article>**)
- 2- Place the "social media" heading and list of links inside an element that represents a section of content that's indirectly related to the main content of the page, (using **<aside>**)
- 3- Finally, place the **h2** and articles inside an element that represents the main content of the **<body>** of the page.

Using Multiple **<header>** and **<footer>** Elements

You learned that the **footer** element typically includes information relevant to its containing section, such as copyright details, author information, or navigation. While a **footer** directly within the **body** element or a wrapper **div** serves as the footer for the entire page, you can use multiple **footer** elements on a single page. These can be nested within other elements like **main**, **section**, or **blockquote** to provide specific footer information for those sections. For instance, we used a **footer** within a **blockquote** to link the quote to its author and source. Similarly, we can add footers inside articles to include information specific to each article, like the publication date. For example, a footer element can be added below the 'Read more' link in each article, stating the publication date.

Like footers, multiple **header** elements can be used in a page, each representing the introductory content of a specific section. It's common to include an article's title, author, date, or category within a header. This approach involves wrapping article titles in **header** tags and including author names in paragraphs within these headers.

The same concept applies to the main Article page, where the **header** element can encompass the article title, author name, and publication date. This semantic markup ensures clear communication not only with browsers and computers but also with humans, such as other developers who may read and work with your code.

The site is now set to incorporate images, an essential web component, which you'll learn to add in the next stage. Images attract users and guide them through the information. You will also continue exploring semantic markup, including text formatting elements that add special meaning to content.

Images, Text and Links

File Paths

Understanding relative file paths is crucial for correctly linking resources and navigating within your website's structure.

In web development, understanding how to add images and create links involves knowing their locations within your site's folder structure. For instance, you might want to add an image from an "images" folder or link to a page in a "books" subfolder inside a "products" folder.

To achieve this, you use file paths in your HTML, which describe the location of a file within the site's folder structure. Unlike absolute URLs that include the full web address, relative URLs are used for linking to files within the same site, specifying paths relative to the location of your current HTML document.

For example, to link to an "article.html" page, you simply include the path to the filename in the link's **href** value. This method is known as using relative URLs.

The root directory is the top-level folder containing all other files and folders. When all files are in the root directory, linking is straightforward; you just use the filename. But for sites organized into different folders, such as separate folders for articles and images, file paths must be adjusted accordingly.

When linking to files in subdirectories, include the folder name followed by a forward slash and the file name. For example, the path would be "articles/article.html". To link to a file in a parent or root folder from a subdirectory, use "../" in the href value to instruct the browser to move up one directory level.

For deeper directory structures, like a subfolder within a subfolder, adjust the path accordingly. For instance, if you have an "articles/2017/article.html" structure, your path would include both the folder and subfolder names.

Lastly, when linking back to the root directory from nested subfolders, use the appropriate number of "../" to indicate how many levels the browser should go up to reach the root directory.

The next step in your learning journey will involve adding images to your web pages using these relative paths.

See path-1.html

Adding Images

Images are crucial in web design, adding visual appeal and helping convey information effectively. Web designers and developers use images for various purposes, including displaying logos, avatars, photographs, illustrations, and charts. Browsers support multiple image formats, with JPEG, GIF, PNG, and SVG being the most common.

JPEG is a popular format for displaying detailed imagery with many colors, such as photographs. To maintain organization, it's recommended to store images in a separate folder, commonly named "img" or "images."

When adding an image to a web page, use the **** tag, which is an empty element without any child content or a closing tag. The **** tag requires a **src** attribute specifying the image's location. For example, to display an image named "featured.jpg" in an "img" folder, the src attribute would be set as **src="img/featured.jpg"**.

Another essential attribute for images is the **alt** attribute, which provides a text description of the image. This attribute is crucial for accessibility, allowing screen readers to describe the image to users with visual impairments. It also provides context when the image cannot be displayed due to errors, unsupported formats, or when images are disabled in the browser. Writing meaningful alt text is a best practice, for example, **alt="Virtual reality user"**.

Images are displayed in their inherent dimensions, but you can add a **title** attribute to provide additional information. This attribute creates a tooltip when users hover over the image, offering further context or description.

In practice, you might add an image to different sections of your website. For instance, you could include a relevant image in an article by using the **** tag with the appropriate **src** and **alt** attributes. Adjusting file paths based on the image's location within your website's directory structure is crucial, especially when dealing with nested folders.

Overall, the effective use of images enhances the user experience on your website, making it more engaging and accessible.

Captioning Images

In web design, adding images effectively enhances user engagement. To display an image, such as "vr-user.jpg," in a nested structure like the "2023" subfolder within "articles," you need to navigate through the directory structure correctly. This involves moving up from the current subfolders to the root directory and then accessing the "img" folder.

For the "article.html" file in the "2023" subfolder, you add an **** element right below the closing header tag. Since this file is two levels deep from the root, you use two "../" in the file path (**../..**) to navigate up to the root directory, and then specify the path to the image in the "img" folder (**img/vr-user.jpg**). Also, provide descriptive alt text for accessibility, such as "User trying a VR headset."

HTML also allows for adding captions to images using the **<figure>** and **<figcaption>** elements. The **<figure>** element acts as a container for images and their associated captions, while **<figcaption>** provides a caption for the image within the figure. This is similar to how images are often accompanied by captions in magazines, newspapers, and books.

For instance, in the "index.html" file, you can wrap the image tag with **<figure>** tags and add a **<figcaption>** below the image. The text for the caption can be taken from the image's title attribute or any other descriptive text relevant to the image. This not only adds context to the image but also enhances accessibility.

Remember, multiple images sharing the same caption can be nested within a single **<figure>** element. It's encouraged to experiment with adding images and captions to your project using these elements, which can significantly improve the visual appeal and user experience of your web pages.

Exercise:

For the code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Moon</title>
  </head>

  <body>

  </body>
</html>
```

- 1- Inside the **<body>**, display the image moon.jpg located inside a folder named img.

- 2- Provide the browser alternative text describing the image.
- 3- Finally, add a caption that describes the image.

Creating Breaks in Content

In HTML, whitespace such as multiple spaces, tabs, or new lines is condensed into a single space by default. However, there are situations where specific line breaks or thematic divisions in content are necessary, such as in poetry, addresses, or different sections of text.

To create intentional line breaks without starting a new paragraph or heading, you can use the `
` tag. For instance, when marking up an address within an `<address>` element, placing a `
` tag after each line ensures that the address is displayed on separate lines. The `
` tag is an empty element, meaning it doesn't require a closing tag.

The `<hr>` tag, or horizontal rule, represents a thematic break in content, similar to a scene change in a story or a transition between topics within a section. This can be useful for visually and semantically separating different topics within a content section. By default, browsers display `<hr>` as a horizontal line.

Both `
` and `<hr>` tags can be written with or without a trailing slash, and both forms are valid in HTML. However, it's crucial to use these elements appropriately. The `<hr>` tag should not be used merely for aesthetics, like adding a line for decoration, and the `
` tag should not be used excessively to create space between lines of text. These styling aspects are better handled by CSS (Cascading Style Sheets), which you'll learn about in more detail in later courses.

See path-3.html

Add Meaning to Words with Text Level Elements

In HTML, formatting text to highlight its importance or emphasis is crucial. Semantic elements like ``, `<small>`, ``, and `` are used for this purpose.

- The `` tag signifies strong importance or urgency. The text within `` tags typically appears in bold, making it stand out. This tag can be used within headings, paragraphs, etc., to highlight key parts of the text. Although the default style is bold, CSS can be used to alter its appearance, such as changing the size or color.
- The `` tag is used for emphasizing text, which by default, is displayed in italic. This emphasis can alter the meaning of a sentence, stressing a particular word or phrase.
- The `<small>` tag is intended for small text such as disclaimers, legal notices, or copyright information. It should be used for short spans of text and not for reducing the size of long text passages.
- The `` tag is used to group text for styling purposes. Unlike ``, ``, and `<small>`, which have default styles, `` does not change the text's appearance

without CSS. It's an inline element, meaning it can be placed within a line of text without causing a break.

For instance, in a contact section, using **** tags can make elements like "email," "phone," and "address" stand out, aiding the scanability of information. In a header, **** tags around a word like "simple" can emphasize that the blog is user-friendly. And in the footer, using **<small>** tags for the copyright text is appropriate.

The difference between **** and **<div>** is that the latter is a block-level element, typically used for larger content blocks and creating breaks before or after the content. In contrast, **** is inline, suitable for styling parts of text within a block.

These elements enhance the semantic structure and readability of web content, ensuring the right emphasis and style are conveyed.

See path-4.html

Exercise:

For the code:

```
<!DOCTYPE html>
<html>

<head>
  <title>HTML Text</title>
</head>

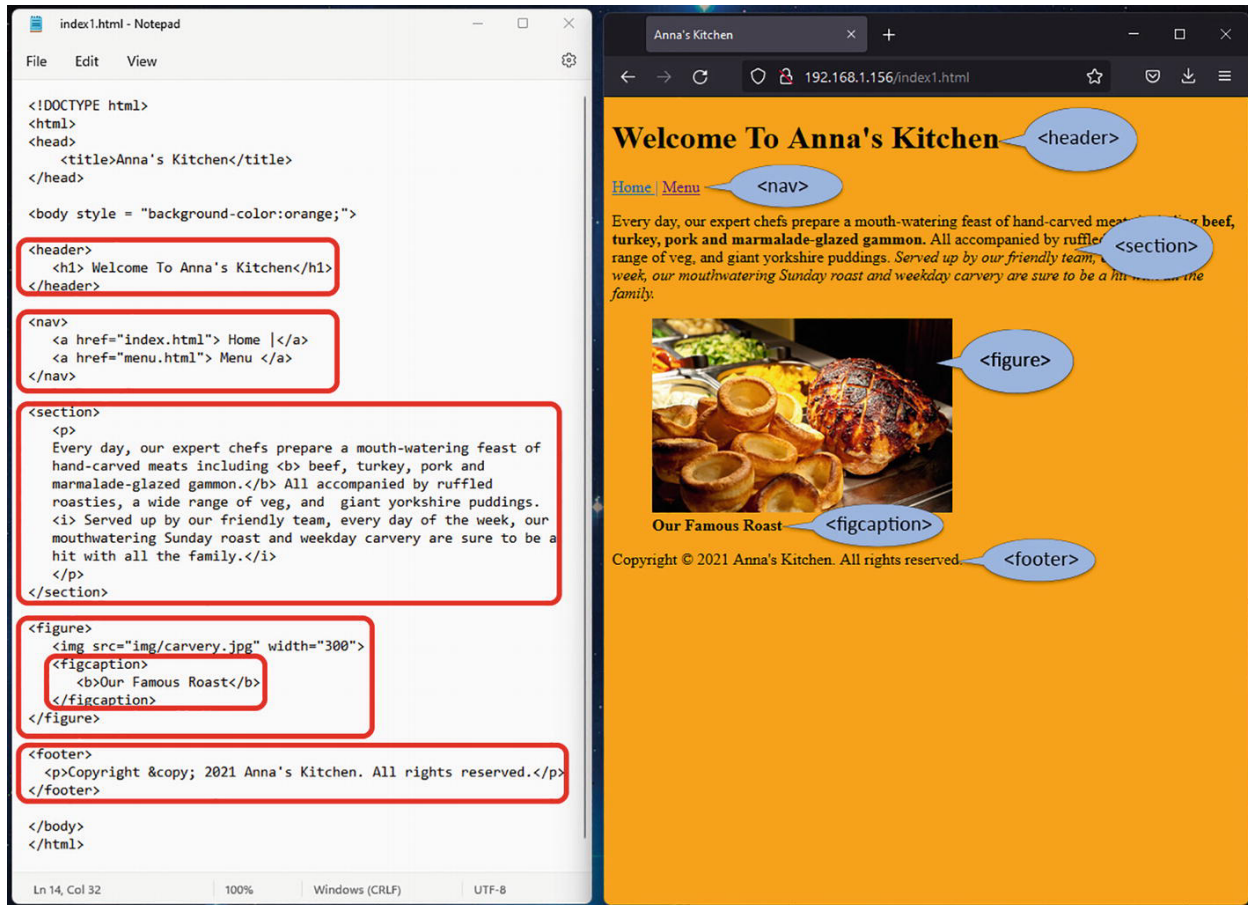
<body>
  <p>Mike's favorite course is HTML Basics</p>
  <p>
    Mike T. Frog
    100 Lilypad Way
    Portland, OR 97227
  </p>
</body>

</html>
```

- 1- Add line breaks to the address in the second paragraph.
- 2- Give the text "Mike T. Frog" strong importance, using the element that displays text in bold.
- 3- Use the element that displays text in italic to add emphasis to the words "HTML".

Summary

The following image shows some of the most common HTML elements that were described in this section:



Source: The Absolute Beginner Guide to HTML and CSS, Kevin Wilson, O'Reilly.

- When using the local Abyss Server on your computer, save documents to C:\Abyss Web Server\htdocs.
- To access the website on your computer with a browser, use `http://127.0.0.1`.
- Main heading style: `<h1>...</h1>`
- Subheading style: `<h2>...</h2>`
- Minor heading style: `<h3>...</h3>`
- Bold text: `...`
- Italic text: `<i>...</i>`
- Paragraph text: `<p>...</p>`
- Use the `img` element to add an image: ``.
- Use the anchor element to define hyperlinks: ``.

Comments

An HTML **comment** is a portion of the document that is not displayed by the browser. A comment begins with the <!-- character sequence and ends with the --> character sequence. Web developers sometimes use comments to leave notes for themselves or others, or to tell the browser to ignore part of the document. *A common error is trying to put a comment inside of another comment.*

Validating Your Website

You can check the integrity of your html code using the following link.

<https://validator.w3.org/>

With the knowledge you have now, you may not be able to properly style your webpages, but if you are given the source code of a webpage, you must be able to modify the contents to your choices and have a way to create a customized nice looking page of yours. There are many places you can find templates for web pages. You can try the following link and play around with one.

<https://www.html.am/templates/>

Exercise:

Using the templates in the above link, find a template and modify the contents with a page about a product.