5a Theory JS API pages 77 - 88 HTML Basics

Nicholas Bruzzese

July 26, 2025

HTML, or HyperText Markup Language, serves as the foundation for web pages, enabling the creation of structured content that web browsers interpret and display. The term "HyperText" refers to text with hyperlinks, which are clickable connections to other web pages or resources. A markup language uses tags to annotate text, providing instructions to browsers on how to present or interact with content, unlike plain text, which lacks formatting instructions. This lecture explores the essentials of HTML, focusing on creating basic web pages, using text editors, and understanding key HTML components such as elements, tags, attributes, and document structure, tailored for a young learner with some JavaScript experience.

1 Text Editors for Writing HTML

To write HTML, a text editor creates plaintext files containing only text without formatting like fonts or colours found in word processors. Plaintext ensures the file contains only code and content, avoiding hidden formatting that could interfere with browser interpretation. Sublime Text, a recommended text editor, is cross-platform, compatible with Windows, Mac OS, and Linux. It offers a free trial but requires a paid license after extended use. Free alternatives include Gedit (cross-platform), Notepad++ (Windows), and TextWrangler (Mac OS), all suitable for HTML. Installation instructions for Sublime Text are available at its official website, with similar steps applying to other editors due to their simplicity.

Sublime Text provides syntax highlighting, colouring different code parts to enhance readability. For instance, JavaScript strings might appear green, and keywords like `var` might be orange. This visual aid helps identify code components quickly. Users can select colour schemes, such as the IDLE scheme, via the editor's preferences menu.

2 Creating a Basic HTML Document

An HTML document is a plaintext file with a `.html` extension, such as `page.html`. In Sublime Text, create a new file and save it with the `.html` extension, for example, on the desktop. Consider the following JavaScript-related HTML example:

```
    <html>
    <body>
    <h1>Hello world!</h1>
    <p>My first web page.</p>
    </body>
    </html>

```

After saving, open the file in a browser like Chrome using "Open File" and selecting `page.html`. The browser displays the content locally, not on the internet. The `<h1>` tag creates a large, bold heading, "Hello world!", while the `<p>` tag creates a paragraph, "My first web page.", showing how HTML tags control presentation, which can later integrate with JavaScript for interactivity.

3 HTML Elements and Tags

HTML documents use elements defined by start and end tags. A start tag, like `<h1>`, begins an element, and an end tag, like `</h1>`, closes it, with content in between. For example, in `<h1>Hello world!</h1>`, "Hello world!" is the content, displayed as a heading. Similarly, `<p>My first web page.</p>` defines a paragraph. Tags use angle brackets `<>`, with end tags including a forward slash `/`, a structure familiar to JavaScript programmers working with HTML.

4 Heading Elements

HTML provides six heading levels, `<h1>` to `<h6>`, each indicating importance and visual size. The `<h1>` tag is the largest, used for main titles, while `<h6>` is the smallest, for minor subheadings. For example:

```
    <h1>First-level heading</h1>
    <h2>Second-level heading</h2>
    <h3>Third-level heading</h3>
    <h4>Fourth-level heading</h4>
    <h5>Fifth-level heading</h5>
    <h6>Sixth-level heading</h6>

```

These headings appear in decreasing size, organising content hierarchically, useful for structuring web pages that JavaScript can manipulate.

5 The Paragraph Element

The `<p>` element defines a paragraph, displaying text with spacing above and below. For example:

```
    <p>Hello world!</p>
    <p>My first web page.</p>
    <p>Let's add another paragraph.</p>

```

This creates three paragraphs, each on a new line with spacing, enhancing readability, a feature JavaScript can leverage for dynamic content updates.

6 Whitespace and Block-Level Elements

In HTML, whitespace (spaces, tabs, newlines) collapses into a single space in the browser. Without tags, text like "Hello world! My first web page. Let's add another paragraph." appears as one line. Block-level elements, like `<h1>` and `<p>`, create separate content blocks, starting and ending on new lines, ensuring clear separation, which JavaScript can target for manipulation.

7 Inline Elements

Inline elements, such as `<em>` and `<strong>`, format text within a line without creating new lines. The `<em>` element italicises content, and `<strong>` makes it bold. For example:

```
    <p>My <em>first</em> <strong>web page</strong>.</p>

```

To apply both styles, nest elements correctly, ensuring inner tags are fully contained within outer tags:

```
    <p>Let's <strong><em>add another</em></strong> paragraph.</p>

```

Incorrect nesting, like `<strong><em>text</strong></em>`, can cause issues, a concept relevant when JavaScript interacts with HTML elements.

8 Structure of a Full HTML Document

A complete HTML document includes additional elements:

```
    <!DOCTYPE html>
    <html>
    <head>
    <title>My first proper HTML page</title>
    </head>
    <body>
    <h1>Hello world!</h1>
    <p>My first web page.</p>
    </body>
    </html>

```

The `<!DOCTYPE html>` declaration signals an HTML document. The `<html>` element is the root, containing `<head>` and `<body>`. The `<head>` holds metadata, like `<title>`, which sets the browser tab's text. The `<body>` contains visible content. Sublime Text indents nested elements, reflecting hierarchy, aiding JavaScript developers in structuring pages.

9 HTML Hierarchy

HTML elements form a hierarchy, like an upside-down tree or nested boxes, with `<html>` as the root, containing `<head>` and `<body>`. The `<head>` includes `<title>`, and `<body>` holds elements like `<h1>` and `<p>`. This structure guides browsers and JavaScript in interpreting and manipulating the document.

10 Adding Hyperlinks

Hyperlinks, created with the `<a>` (anchor) element, enable navigation. For example:

```
1     <p><a href="http://xkcd.com">Click here</a> to read some
    excellent comics.</p>
2
```

The `href` attribute specifies the URL, like `http://xkcd.com`. The text, "Click here", is the clickable link. The `title` attribute displays hover text:

```
1     <a href="http://xkcd.com" title="xkcd: Land of geeky comics">
    Click here</a>
2
```

Attributes resemble JavaScript object key-value pairs, a familiar concept for JavaScript programmers adding interactivity to links.

11 Practical Application

Create a `links.html` file with a similar structure to `page.html`, but with a new title, heading, and three `<p>` elements, each containing an `<a>` element linking to a favourite website, including `href` and `title` attributes, practicing skills for JavaScript-enhanced pages.

12 Conclusion

HTML structures web content using elements, tags, and attributes. Text editors like Sublime Text, with syntax highlighting, facilitate HTML writing. Block-level elements like `<h1>` and `<p>` organise content, while inline elements like `<em>` and `<strong>` format text. A full HTML document includes a doctype, `<html>`, `<head>`, and `<body>`, forming a hierarchy. Hyperlinks with `<a>` elements enable navigation using `href` and `title` attributes. These fundamentals support JavaScript integration for interactivity, with further learning available from resources like the Mozilla Developer Network, Codecademy, and Mozilla Webmaker.