Introduction to HTML

Abdulla Ebrahim Subah

October 24, 2024

Introduction to HTML

What is HTML?

- **HTML** stands for **HyperText Markup Language**.
- It is the standard language for creating web pages.
- HTML describes the structure of a webpage.

Basic Structure of an HTML Document

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>This is a Heading</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

Key Elements

- **`<!DOCTYPE html>`**: Defines the document type and version.
- **`<html>`**: The root element of an HTML page.
- **`<head>`**: Contains meta-information about the document.
- **`<title>`**: Sets the title of the webpage.
- **`<body>`**: Contains the content of the webpage.

The `<head>` and `<body>` Tags

- **`<head>` Tag**:
  - Contains metadata and links to resources.
  - Includes elements like `<title>`, `<meta>`, `<link>`, and `<style>`.
  - Does not display content directly on the webpage.
- **`<body>` Tag**:
  - Contains the content of the webpage.
  - Includes text, images, links, and other media.
  - Everything inside is rendered in the browser window.

HTML Tags, Elements, and Attributes

Tags

- Tags are used to create elements.
- They usually come in pairs: an opening tag and a closing tag.
- Example: `<p>...</p>`

Elements

- An element consists of a start tag, content, and an end tag.
- Example: `<h1>This is a heading</h1>`

Attributes

- Attributes provide additional information about elements.
- They are always included in the opening tag.
- Example: `<a href="https://example.com">Visit Example</a>`

HTML Syntax

- **Tags**: Enclosed in angle brackets, e.g., `<tagname>`.
- **Elements**: Consist of a start tag, content, and an end tag.
- **Nesting**: Elements can contain other elements.
- **Attributes**: Key-value pairs in the opening tag, e.g., `id="header"`.
- **Self-closing Tags**: Some tags don't require a closing tag, e.g., `<br />`.
- **Case Sensitivity**: HTML tags and attributes are not case-sensitive, but lowercase is recommended.

HTML Editors

- **Visual Studio Code**: Popular, extensible, and free.
- **Sublime Text**: Lightweight and fast with powerful features.
- **Notepad++**: Simple and efficient for Windows users.
- **Vim**: Highly configurable and powerful for advanced users.
- **Nano**: Simple, easy-to-use terminal-based editor.
- **Any Text Editor Will Work**

HTML Standard and Resources

- **HTML Standard**: WHATWG HTML Standard
- **HTML Tags List**: W3Schools HTML Tags Reference

Famous HTML Tags with Examples

Headings

```html
<h1>Main Heading</h1>
<h2>Subheading</h2>
```

Paragraphs

```html
<p>This is a paragraph.</p>
```

Links

```html
<a href="https://example.com">Visit Example</a>
```

Images

```html
<img src="image.jpg" alt="Description">
```

Lists

- **Ordered List**

```html
<ol>
  <li>First item</li>
  <li>Second item</li>
</ol>
```

- **Unordered List**

```html
<ul>
  <li>First item</li>
  <li>Second item</li>
</ul>
```

Div and Span

- **Div**

```html
<div>A block-level container.</div>
```

- **Span**

```html
<span>An inline-level container.</span>
```

HTML Forms and Inputs

HTML Form Basics

- Forms allow users to input data and submit it to a server.
- Basic structure:

```html
<form action="/submit" method="post">
  <!-- Form elements go here -->
</form>
```

Common Form Input Types

- Text: `<input type="text">`
- Password: `<input type="password">`
- Radio: `<input type="radio">`
- Checkbox: `<input type="checkbox">`
- Submit: `<input type="submit">`

HTML5 Input Types

- Email: `<input type="email">`
- Number: `<input type="number">`
- Date: `<input type="date">`
- Color: `<input type="color">`
- Range: `<input type="range">`

Input Attributes

- `name`: Specifies the name of an input element
- `value`: Specifies the initial value of an input element
- `placeholder`: Specifies a hint that describes the expected value
- `required`: Specifies that an input field must be filled out
- `disabled`: Specifies that an input field should be disabled

Example: Various Input Types

```html
<form>
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required><br>

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

  <label for="password">Password:</label>
  <input type="password" id="password" name="password"><br>

  <label for="age">Age:</label>
  <input type="number" id="age" name="age" min="0" max="120"><br>

  <input type="submit" value="Submit">
</form>
```

HTML5 Form Validation

- HTML5 introduced built-in form validation
- Reduces the need for JavaScript validation
- Provides a better user experience

HTML5 Validation Attributes

- `required`: Field must be filled out
- `minlength` and `maxlength`: Minimum and maximum text length
- `min` and `max`: Minimum and maximum values for numerical inputs
- `pattern`: Defines a regular expression the input must match

Example: HTML5 Form Validation

```
<form>
  <label for="username">Username (4-8 characters):</label>
  <input type="text" id="username" name="username"
         required minlength="4" maxlength="8" pattern="[A-Za-z0-9]+"><br>

  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required><br>

  <label for="age">Age (18-99):</label>
  <input type="number" id="age" name="age" min="18" max="99" required><br>

  <input type="submit" value="Submit">
</form>
```

Other Form Elements

- `<textarea>`: Multi-line text input
- `<select>` and `<option>`: Dropdown list
- `<button>`: Clickable button (can be used instead of `<input type="submit">`)
- `<fieldset>` and `<legend>`: Group related form elements

Example: Additional Form Elements

```
<form>
  <fieldset>
    <legend>Personal Information:</legend>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"><br>

    <label for="bio">Bio:</label>
    <textarea id="bio" name="bio" rows="4" cols="50"></textarea><br>

    <label for="country">Country:</label>
    <select id="country" name="country">
      <option value="usa">USA</option>
      <option value="uk">UK</option>
      <option value="canada">Canada</option>
    </select><br>

    <button type="submit">Submit</button>
  </fieldset>
</form>
```

Accessibility in Forms

- Use `<label>` elements to associate labels with form controls
- Provide clear instructions and error messages
- Use ARIA (Accessible Rich Internet Applications) attributes when necessary
- Ensure keyboard navigation works properly
- Test with screen readers and other assistive technologies

HTML Forms and Attributes

Basic Form Structure

```
<form action="/submit" method="post">
  <!-- Form elements go here -->
</form>
```

Important Form Attributes

- `action`: Specifies where to send the form data when submitted
- `method`: Defines how data is sent (GET or POST)
- `enctype`: Specifies how form data should be encoded
- `name`: Gives the form a name for reference
- `target`: Specifies where to display the response after submission

The `action` Attribute

- Defines the URL where form data is sent upon submission
- Can be a relative or absolute URL
- If omitted, form submits to the current page

Example:

```
<form action="/submit-form">
```

```html
  <!-- Form elements -->
</form>
```

The `method` Attribute

- Specifies how form data should be sent to the server
- Two main values: `GET` and `POST`

1. `GET`:
   - Appends form data to the URL
   - Used for non-sensitive data
   - Limited amount of data can be sent
2. `POST`:
   - Sends form data in the body of the HTTP request
   - Used for sensitive data (e.g., passwords)
   - No limit on amount of data

Example:

```html
<form method="post">
  <!-- Form elements -->
</form>
```

Other Important Attributes

- `enctype`: Specifies how form data should be encoded
  - Default: `application/x-www-form-urlencoded`
  - For file uploads: `multipart/form-data`
- `autocomplete`: Enables/disables browser autocomplete
- `novalidate`: Disables browser's automatic validation

Example:

```html
<form action="/upload" method="post" enctype="multipart/form-data">
  <input type="file" name="photo">
  <input type="submit" value="Upload">
</form>
```

Form Security Considerations

- Use HTTPS for secure data transmission

- Implement server-side validation
- Use POST for sensitive data
- Be cautious with user-supplied data to prevent XSS attacks
- Implement CSRF protection for sensitive actions

Semantic HTML

What is Semantic HTML?

- Uses tags that convey meaning about their content
- Describes the structure and purpose, not just appearance
- Improves accessibility, SEO, and code readability

Common Semantic HTML Tags

- `<header>`: Introductory content or navigation
- `<nav>`: Navigation links
- `<main>`: Main content of the document
- `<article>`: Self-contained composition
- `<section>`: Thematic grouping of content
- `<aside>`: Content tangentially related to the main content
- `<footer>`: Footer of a document or section

Example: Non-Semantic vs Semantic HTML

Non-Semantic:

```html
<div id="header">
  <h1>My Website</h1>
  <div id="nav">
    <a href="#home">Home</a>
    <a href="#about">About</a>
  </div>
</div>
<div id="main">
  <div class="article">
    <h2>Article Title</h2>
    <p>Article content...</p>
  </div>
```

```html
</div>
<div id="footer">
  <p>&copy; 2024 My Website</p>
</div>
```

Semantic:

```html
<header>
  <h1>My Website</h1>
  <nav>
    <a href="#home">Home</a>
    <a href="#about">About</a>
  </nav>
</header>
<main>
  <article>
    <h2>Article Title</h2>
    <p>Article content...</p>
  </article>
</main>
<footer>
  <p>&copy; 2024 My Website</p>
</footer>
```

Benefits and Best Practices

Benefits: - Improved accessibility for screen readers - Better SEO (Search Engine Optimization) - Easier to read and maintain code

Best Practices: 1. Use HTML5 semantic elements when possible 2. Choose the most specific tag for the content 3. Use headings (`<h1>` to `<h6>`) logically 4. Avoid `<div>` when a semantic element exists 5. Use ARIA roles when necessary for accessibility