Extra notes - Markup Languages

- Dr Nick Hayward

HTML - Intro

A brief introduction to HTML.

Contents

Intro

HTML is an acronym for *HyperText Markup Language*. It is a simple way to structure visual components of a website or web application.

As a programming language may rely on keywords, HTML also uses keywords, or element tags as they are better known, that follow a defined syntax.

However, instead of creating a working desktop application, similar to a more traditional programming language such as C or Java, HTML helps us to create web pages and web applications that web browsers, such as Chrome or Firefox, may render for viewing.

In a traditional programming language, an error or bug can keep the entire program from running or make it calculate incorrectly. In HTML, an error can stop a web page from rendering, but more likely it will simply cause the page itself to be rendered incorrectly.

If you are interested in understanding the core of web page designing, you need to understand at least the basics of using HTML.

Element syntax

HTML is constructed using elements and attributes, which are embedded within an HTML document. Elements should adhere to the following,

- start with an opening element tag, and close with a matching closing tag
  - names may use characters in the range **0-9**, **a-z**, **A-Z**
- content is, effectively, everything between opening and closing element tags
- elements can contain empty or void content
- empty elements should be closed in the opening tag
- most elements permit attributes within the opening tag

So, an element's *start* tag adheres to a structured pattern, which is exactly as follows,

1. a **<** character
2. tag name
3. optional **attributes**, which are separated by a space character
4. optional space characters (one or more...)

5. optional **/** character, indicating a **void** element
6. a **>** character to close the element tag

For example,

```
<!-- opening element tag -->
<div>
<!-- void element -->
<br />
```

An element's *end* tag also adheres to a pattern, again defined as follows,

1. a **<** character
2. a **/** character
3. element's tag name (i.e. name used in matching start tag)
4. optional space characters (one or more...)
5. a **>** character

For example,

```
</div>
```

As noted above, **void** elements, such as `<br />` or `<img />`, do *not* specify end tags.

HTML, and its stricter derivation XHTML, can be written to follow the patterns and layouts of XML. Similar to XML, HTML elements can also be nested with a parent, child, sibling &c. relationship within the overall tree data structure for the document.

So, as the HTML page is loaded by a web browser, the HTML *DOM* (document object model) is created. This is, basically, a tree of objects that constitutes the underlying structure of the rendered HTML page. The DOM gives us an API (application programming interface), which provides a known way of accessing and manipulating the underlying elements, attributes, and content.

This will become particularly useful and interesting as a developer adds JavaScript to a project's development. With traditional JavaScript, excluding AngularJS or React for example, we may directly manipulate the DOM as required by our given application.

Attribute syntax

Attributes in HTML also follow the same design pattern as other markup languages, such as XML. They provide additional information to the parent element, are placed in the opening tag of the element, and follow the standard syntax of name and value pairs.

There are many different permitted legal attributes in HTML, and four common names that are permitted within most HTML elements.

These attributes include `class`, `id`, `style`, and `title`.

For example,

- class
  - specifies a classname for an element
- id
  - specifies a unique ID for an element
- style
  - specifies an inline style for an element

- title
  - specifies extra information about an element
  - can be displayed as a tooltip by default

NB: you cannot have two or more attributes with the same name, regardless of case, on the same element start tag.

Values for attributes may include text and character references (e.g. `&#228`, which is a Latin `a` with two dots, a German umlaut for example). There are also restrictions on content depending on whether we use double quotes, single quotes &c. to encapsulate the value itself. So, specific to HTML syntax, attributes can be specified in four different ways,

- empty attribute syntax
  - `<input disable>`
- unquoted attribute-value syntax
  - `<input value=yes>` (NB: if value followed by `/`, there must be at least one space character after the value and before the `/` character)
- single quoted attribute-value syntax
  - `<input type='checkbox'>`
- double quoted attribute-value syntax
  - `input title="hello">`

For each of the above, there are some further specific restrictions. Many of these are common-sense implementations, and further details can be found in the W3 Docs.

Character encoding

As we add text to our elements, and values for our attributes, it must consist of defined **Unicode** characters. For example,

- The Unicode Consortium
- Unicode Information
  - Unicode examples - many, many examples...

As with most things, there are some exceptions. These may include the following,

- attribute values must not contain `U+0000` characters
  - e.g.
    `U+0000 (NULL), U+0022 (QUOTATION MARK, "), U+0027 (APOSTROPHE, '), U+003E (GREATER THAN, >), U+002F (FORWARD SLASH, /), and U+003D (EQUALS, =) characters`
  - e.g W3C recommendations - 8.1.2.3
- characters must not contain permanently undefined Unicode characters
- and it must not contain control characters other than *space* characters
  - Space - `U+0020`
  - Tab - `U+0009`
  - Line feed - `U+000A` (Linefeed means to advance downward to the next line)
  - Form feed - `U+000C` (Form feed means advance downward to the next "page")
  - Carriage return - `U+000D` (Carriage return means to return to the beginning of the current line without advancing downward)

So, we basically end up with the following definable types of text for content &c.

- normal character data
  - this includes standard text and character references
  - cannot include non-escaped `<` characters

- replaceable character data
  - includes elements for `title` and `textarea`
  - allows text, including non-escaped `<` characters
  - character references
    - a form of markup for representing single characters
    - e.g. a dagger represented as `&dagger;` or `&#8224;` or `&#x2020;`
    - e.g. copyright symbol `&#169`
    - lots of examples, W3 - Character Ref.

Doctype - HTML5

The `DOCTYPE` is a special instruction to the web browser concerning the required processing mode for rendering the document's HTML.

The `doctype` is a required part of the HTML document, and is the first part of our HTML document. As such, it should always be included at the top of a HTML document. For example,

```
<!DOCTYPE html>
```

or

```
<!doctype html>
```

This is the doctype we add for HTML5 rendering. Therefore, this is not a HTML element, it simply tells the browser the required version of HTML for rendering the current page.

References

- MDN
  - HTML developer guide
- Unicode
  - The Unicode Consortium
  - Unicode Information
  - Unicode examples
- W3 Docs
  - Attribute Syntax
  - Element Syntax