Extra notes - HTML - Basics

- Dr Nick Hayward

A brief introduction to some of the basics of HTML.

Contents

Intro As we design and build out our HTML pages, there are some basics that we may follow for each page, site, and application.

`<head>` element The `<head>` element is an important, and useful, part of a HTML document's metadata. The `<head>` element allows us to set metadata for a HTML page, which can either be customised just for that page or replicated as a site-wide implementation.

add some metadata Within the `<head>` section of our web page, we can add numerous additional elements. For example, in addition to CSS information, we may add similar links and code for JavaScript.

To link to an external JavaScript file we use the `<script>` element with required attributes such as `type` and `src` . The specific required attributes vary from HTML4 to HTML5. For example,

```
<!-- HTML4 and XHTML -->
<script type="text/javascript" src="script.js"></script>
<!-- HTML5 -->
<script src="script.js"></script>
```

So, in our example, we are now including a JavaScript file for use within our HTML page. This means that we can use JavaScript functions, which will be available for use with defined IDs and classes in the page. i.e. our defined selectors within the DOM. We may also choose to add such JavaScript references to the foot of a given page instead of the more traditional `<head>` element.

We can also set a title for our page, which will be shown as the page title in a browser tab or window heading. This is simply a `<title>` element with text added as the element content.

define a base address We may also set a default **base** address for all relative URLs in links within our HTML. This means we could specify the base directory once in the `<head>` section, and then add the filename or image location **relative** to the base address, instead of having to write out the base each time as well. You can override this base address at any time in a link, simply by expressing the full, or **absolute**,

URL. Also, you can only add one base address per page, and it must be listed in the `<head>` element. For example,

```
<base href="/media/images/" target="_blank">
```

The `<base>` tag provides a base URL for all links in a document, which can be useful for a large number of links with the same base location, such as an image gallery. It's considered good practice to add this element at the top of the `<head>` element, so we may then reference it later in the `<head>` element, if necessary. However, you have to ensure that other links use the full URL, or they will similarly inherit this base URL. Within the `<body` of our HTML document, we can use it as follows,

```
<img src="image.jpg">
<a href="http://www.flickr.com">Flickr</a>
```

Additional metadata for the web page can also be added to the `<head>` section.

We can add author information, a description, keywords to help indexing, revision information, language information and so on.

```
<head>
  <meta charset="utf-8">

  <title>Sample...</title>
  <meta name="description" content="sample metadata">
  <meta name="author" content="COMP424">

  <link href="style.css" rel="stylesheet">
  <script src="script.js"></script>

</head>
```

example usage

`<body>` **element** The body element allows us to now start designing and adding some structure for rendering and presenting our web page. We could start outlining our new HTML document by adding some headings.

In HTML, these are defined by default as `<h1>` to `<h6>` , with *1* the largest all the way to the smallest *6*.

Headings are meant to be used as headings, and not simply for bold &c. By default, a browser automatically adds a margin before and after each heading. This can then be customised with the app's CSS.

We can then add a `<p>` element to the `<body>` to allow us to add some simple text within a paragraph.

If we then wanted to add a line break, we could add the `<br>` element. Note that if we were writing strict XHTML, the tag should be `<br />` to define it as a void element.

We could also divide sections of our paragraph using a simple `<hr>` element, which adds a horizontal line or rule to the rendered page. However, it's becoming less common to see such a division. It often implies a rendering division instead of a defined structural divide and break.

HTML is also similar to many programming languages in allowing us to clearly and easily comment throughout our document. We can simply add a comment as follows,

```
<!-- comment... -->
```

linking or hyperlinking Linking, or hyperlinking, is a nearly ubiquitous action within web pages and web design.

Even within a single page app there will normally be some semblance of links and internal anchors.

So, linking within HTML can be considered in a number of different contexts. For example,

1. linking to an external site
2. linking to another page within the same site
3. linking different parts of the same page

Within web design and HTML, it is standard to add links to text and images embedded within a page.

To add any link we use the anchor element tag, which may be used to achieve any of the different contexts already outlined.

So, to link to an external page we need to use the anchor element plus some additional attributes.

The default attribute is named `href` . This is added to the anchor element, and its value should be set to the link required, as demonstrated in our examples.

We can use this construction for internal links within our website, and external links to other pages and websites.

However, it is better to set the value of the `href` attribute as relative to the root directory of the installed website or domain name for internal links.

Link addresses may also be specified relative to directory depth, which follows the standard Unix practice for traversing directories.

We can also add anchors and links for traversing within the same HTML document. To achieve this practice, we need to specify our anchor using a `<name>` attribute, or preferably `<id>` in XHTML, and then link to that anchor using the standard anchor element with `href` attribute.

We can also reference anchors from other pages by affixing the name reference to the url for the page itself as a value within a standard 'href' attribute.

linking - example usage Links in HTML may be specified as follows,

```
<!-- external link -->
<a href="http://www.google.com/">Google</a>
<!-- email link -->
<a href="mailto:name@email.com">Email</a>
<!-- internal page link -->
<a href="another_page.html">another page</a>
<!-- define internal anchor - using name attribute -->
<a name="anchor">Internal anchor</a>
<!-- define internal anchor - using ID attribute -->
<a id="anchor">Anchor</a>
<!-- internal anchor link -->
<a href="#anchor">Visit internal anchor</a>
<!-- internal anchor link on another page -->
<a href="/another_page.html#anchor">Visit internal anchor</a>
<!-- internal anchor link on a page on an external site -->
<a href="https://www.test.com/test.html#anchor">Visit internal anchor on external site</a>
```

linking - extras We can also add further attributes to the anchor element including,

- standard HTML global attributes such as `class, id, lang, style, title...`

and

- anchor element optional attributes such as `target, href, name...`

We've already discussed the `href` and `name` attributes, but `target` is another useful attribute for links.

The `target` attribute specifies where a link should be opened relative to the current viewed browser window.

For example,

- `_blank` = open the link in a new window or tab (dependent upon new window settings in the browser preferences)
- `_self` = same frame as it was clicked (default option) (actual frames used for structuring)
- `_parent` = opens within the parent frameset of the current link regardless of frame location
- `_top` = opens the link in the same window as the current page

If you do not intend to use frames then you should only really concern yourselves with `_blank` and `_top` .

The others will still work regardless of frames, but you should be aware of the original intent.

`<img>` **- adding images** By using the HTML element `<img>` we are able to add a link to an image, a placeholder of sorts, both relative and absolute, within our HTML page.

The `<img>` element also requires attributes to function as commonly expected.

The `src` attribute works like the `href` attribute in the anchor link element. It allows us to include the filename of the required image, and, where applicable, the location of that file.

Our example would look for an image file called `image.jpg` at the same directory level as the current HTML page. If the image was located elsewhere, our `src` value would need to reflect this as well unless we had set a base URL for the page.

We can also use some optional attributes with our `<img>` element.

These include,

- `class` (specifies css class for styling)
- `id` (unique reference id)
- `alt` (renders alternative text, defined by the developer, if the image is unable to be loaded)
- `title` (will show a tooltip message for the image, again specified by the developer)
- `width` & `height` allow a developer to specify a size for the space to hold an image. The image is resized to fit the given dimensions. If the image is not found, the alt text will be shown in the specified sized space. The dimensions of the sized space will be maintained within the html.

We can also nest images within anchor links, thereby turning the image itself into a link.

Images can also be used to create image maps, which allow us to designate coordinates within a given image where we may attach links and other actions. Basically, we are making designated areas of an image clickable or interactive with internal or external links.

working with tables In HTML, tables allow us to group data in a similar fashion to a spreadsheet or database table.

To add a table you need to start with the `<table>` element. There are then three main elements that act as children to the parent `<table>` element, allowing us to add column headers, table rows, and individual row cells.

To add a row we use the `<tr>` element. Then we can add a column header using the `<th>` element.

Finally, we can add our table cells, nested within table rows, using the table data `<td>` element.

We can also add the usual styling considerations with CSS. So, we could set a border for our table, a different colour for odd rows, header colour and padding and so on. We can also add a caption to our table, which is normally set as the first child element of our table. We could add further headers as needed, as well.

Columns and rows can span multiple cells using either the `colspan` or `rowspan` attribute respectively.

- e.g. 'colspan="4"'

And, we can also nest other elements within a table, such as a link or image.

```
<table>
  <caption>424 - basic test table</caption>
  <tr>
    <th>heading 1</th>
    <th>heading 2</th>
  </tr>
  <tr>
    <td>row 1, cell 1</td>
    <td>row 2, cell 2</td>
  </tr>
</table>
```

tables - example usage

working with lists We can also organise data using lists. In HTML, there are various types of lists available to us. These include,

- unordered list - `<ul>`
- ordered list - `<ol>`
- definition list - `<dl>`

An unordered list is basically the same as using a list of bullet points, with list items represented using the `<li>` element. The character used to display the bullet point can be customised using css, such as a circle, disc, square… The default symbol is normally a black circle.

If we start nesting our unordered lists, then the default list type will now be displayed as a disc. If we nest another unordered list as well, the list type will show a square by default. The list type will then continue as a square for further nested lists.

An ordered list works in the same fashion, but instead of using bullet points it marks each list item with a number, which increments by 1 per descending list item.

You can also customise the list type for each list item. For example, we could change the number to a roman numeral, or letters, uppercase and lowercase….

This can be styled in CSS using the `list-style` declaration. However, the default will be a numerical list.

A definition list allows us to organise a series of items with associated definitions. We may add as many definitions as required, and also nest other html elements within our definition.

lists - example usage

- unordered list `<ul>` , ordered list `<ol>` , definition list `<dl>`
- `<ul>` and `<ol>` contains list items `<li>`

```
<ul>
  <li>...</li>
</ul>
```

```
<ol>
  <li></li>
</ol>
```

- definition list uses `<dt>` for the item, and `<dd>` for the definition

```
<dl>
  <dt>Game 1</dt>
  <dd>our definition</dd>
</dl>
```

working with forms Adding forms to your HTML will often become a common requirement within your website.

Forms are used as a way to capture data input by a user, which can then be processed and actioned by the server to return results. To add a form to our webpage, we start by using the `<form>` element.

We now need to add a way for our user to input some data. So, we add an `<input>` element to our new form.

There are various types of input fields and options available, which can be set using a value for the `type` attribute. Options include, for example

- text
- password
- radio button
- checkbox
- submit

```
<form>
  Text field: <input type="text" name="textfield" />
</form>
```

forms - example usage

References

- MDN
  - HTML developer guide
- W3C
  - HTML Attribute Syntax