HTML - Summary

Basics

- *Tags*
  - HTML tags come in pairs - opening and closing ones.

    ```
    <p>A paragraph</p>
    ```

  - However some tags can can be self-closing (img, hr, br).

    ```
    <img src="some-image.png" />
    ```

  - Some others can have only the opening tag (link, meta etc.)

    ```
    <meta name="description" content="This is a description that shall appear in search results in sea
    ```

- *Element Types*: There are many HTML tags - `a`, `img`, `p`, `hr`, `ol`, `table`, `tr`, `form`, `input` etc.
- *HTML element*: Each use of an HTML type - for example, we can have many HTML Paragraph elements on a page.

  ```
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
  <p>Paragraph 3</p>
  ```

- *Element Relationships*: Elements may have content within.
  - elements within are *children* (the opening and closing tags of a children MUST be nested properly within the opening and closing tags of their respective parents)
  - elements deep within are *descendants*
  - an element within which another exists is its *parent*
  - We similarly define *ancestor*
  - elements that are share a common parent are *siblings*
- *Attributes* - They are used to customize elements
  - are specified in opening tags
  - the values are quoted (double quotes preferred).

    ```
    <tag attribute="value">
    ```

  - Some attributes are common to most types (eg. `id`, `title`, `class`, `name`
  - Some are specific - eg. `href` can be used in `link`, `a`; `src` can be used in `img`, `iframe` etc.; `alt` in `img`
  - The `id` attribute value assigned to an element is to be unique. For example the following is undesirable and should be avoided at all cost.

    ```
    <p id="para">Para 1</p>
    <p id="para">Para 2</p>
    ```

  - The `class` attribute value is used to group a set of elements (and hence can repeat). An element can have multiple classes (Separate by spaces).

    ```
    <p id="x y">Para with classes x and y</p>
    <div id="x z">Division with classes x and z</div>
    ```

- Element types are broadly classified as *block-level* and *inline* (although there are other types too).
  - Block level elements do not allow other elements to their left (of the line they start on) or right (of the line they end on), and by default occupy full width of their parent (They respect all *CSS box model properties* - this is

covered in CSS).
    - Inline elements come next to each other and do not respect many box model properties.
- Browser (user agent) supplies default styling for many HTML types. However as an author, you have full control over element styles.

Basic tags

- The `html` tag can have `head` and `body` (both optional)
- *Paragraph* (`p`) are given margins above and below by the browser.
- Consecutive whitespace characters (spaces, tabs, newlines etc.) are rendered as a single space by the browser
    - `<br />` for line break
    - `<hr />` for horizontal rule (line)
    - ` ` for non-breaking spaces. Use similar *HTML symbols* (like `&copy;` `&gt;` , `&lt;` etc.) for displaying non-printable and special characters.
- *Heading*
    - `h1 - h6` as per heading levels
    - Manipulate styles as may be required using CSS
    - Only one h1 element per page is recommended.
- *Anchor*
    - `a` , can link to documents in local server, anywhere on the internet, or even within the same document.
    - Use `href` along with appropriate URL.
        - URL may be *relative* or *absolute*.
        - `.` refers to folder of the HTML document,
        - `..` refers to the parent of that folder.
        - `/` is the root folder, which is the web root folder in web servers (where the HTTP server is started usually).
        - No prefix of any kind is a relative URL with respect to current folder.
        - Use `#elid` to link to an element with `id="elid"` .
        - Use `target="_blank"` to have document open in new tab window
        - The anchor tag can house
            - text content
            - Images, videos etc. (clickable image, video etc.)
            - Anything in general (other elements too)
- *Image*
    - `img` - show image on local server or anywhere on the internet
    - `src` attribute specifies the URL (path)
    - `alt` attribute provides an alternative description if image cannot be shown - good as fallback/for accessibility.

Displaying data

- *List*
    - `ol` - Ordered list for items with inherent ordering (example - list of months)
    - `ul` - Unordered list for items with no ordering (eg. items in a items inside one's handbag)
    - `li` - Houses an item within `ul` or `ol`
    - Use lists whenever it makes semantic sense - example list of links in navugation menu - styling can always be managed using CSS.
    - `list-style-type` CSS property controls numbering/bullet system (or removes it altogether)
- *Table*
    - `table` tag
    - Can have (optionally) `thead` and `tbody` as children
    - `tr` houses table row
    - `th` for heading row columns (table heading cells - provides some default styling), `td` for normal row (table data cells).
    - `border-collapse: collapse` collapses table and cell borders

- `tr:nth-child(2n)` selector is used for "Zebra-striping" rows

Capturing data - inputs, buttons and forms

- *Input* and related elements
  - Input and related elements (textarea, select etc.) can be used standalone (outside form).
  - Input element `type` attribute decides input type - some possible values are `text`, `password`, `checkbox`, `radio`, `search`, `url`, `email`, `date`, `number`, `file`, `button`)
  - `value` attribute gives default value

    ```
    <input type="text" value="John" />
    ```

  - `placeholder` gives hint text - shows up when input is empty.

    ```
    <input type="text" value="John" placeholder="Your name..." />
    ```

- *Button*

  ```
  <input type="button" value="Button text" />
  ```

  - Can also be of type `reset` (Reset form inputs within a form), `submit` (submits the form within which it exists)
  - `button` element can also be used

    ```
    <button>Button text</button>
    ```

- *Form*

  ```
  <form action="/submission-page" enctype="multipart/form-data" method="post" novalidate>
  ```

  - Data is sent to page requested as query parameters (GET request) or in request body (POST).
  - Querystring looks like this - param and value are *URL encoded* which replaces special characters

    ```
    ?param1=value1&param2=value2&...
    ```

  - `action` is the URL of the page requested on form submission - by default, the same page is requested again
  - `enctype="multipart/form-data"` is used in case of file inpput submission
  - `method="post"` is preferred for submission of form with sensitive data - data sent in HTTP request body.
  - `novalidate` prevents HTML5 form validation by browser (used whn we do validation ourselves using JavaScript).
- Other input attributes
  - Some have been discussed before in the section on inputs
  - `name` is needed to send the user input data in an input element to the server on form submission (name is used as query parameter, and user input is the parameter's value)
  - Checkboxes and radio buttons
    - grouped using a common value for `name`
    - Each can have a unique id, and will have a `value` set
    - Use `checked` to check checbox/radio button

      ```
      <div>
        <label><input type="checkbox" name="interests" value="sports"> Sports</label>
        <label><input type="checkbox" name="interests" value="world news"> World news</label>
        <label><input type="checkbox" name="interests" value="entertainment"> Entertainment</label>
      </div>
      ```

  - `select` has `option`s
    - Display text is content of option elements. If submitted value is different from display names, set `value` for the options

      ```
      <select>
        <option value="sp">Sports</option>
      ```

```
    <option value="wn">World news</option>
    <option value="ent">Entertainment</option>
  </select>
```

- Use `selected` to set default option

- The following cause validation on form submission
  - `required` - marks as mandatory field
  - `pattern` - takes Regular expression pattern as value - user inout is validated against it for match
- `label` is used to provide label (good for styling, accessibility). `for` is set to the `id` of associated input

```
<label for="firstname">First Name</label>
<input type="text" id="firstname" />
```