











HTML Cheatsheet

 Created	@July 9, 2021 6:09 PM
 Created By	 Luke Landau
 Last Edited By	 Luke Landau
 Last Edited Time	@July 11, 2021 5:30 PM
 Note	Created for and by a pack of fighting mongooses (take with mucho salt)
 Stakeholders	
 Status	
 Type	

HTML Basics

- Hyper Text Markup Language provides the framework for webpages
- achieved with `<tag>` elements filled with content, HTML is used to instruct web browsers how to structure it. The best complete resource I've found so far is [MDN Web Docs](#)

```
<!-- This is a comment and will be ignored by the browser -->

<!-- Declare file as HTML to instruct browser handling -->
<!DOCTYPE html>

<!-- This is the root tag, containing all other HTML code -->
<html>

<h1>This Is The Webpage Title, And Should Only Be Used Once Per Page</h1>
<h2>This is the largest subheading on the page</h2>
<h3>This is the second largest subheading on the page</h3>
<h4>This is the third largest subheading on the page</h4>
<h5>This is the fourth largest subheading on the page</h5>
<h6>This is the smallest subheading on the page</h6>
<p>This is a paragraph</p>
```

```

<p>This is hacky paragraph with a number of <br> line breaks inside. If you ever
do this <br> Erebus, the god of darkness and coding errors, will take you</p>

<div>
  <h2>This is subheading inside a division</h2>
  <p>A division is a customisable container used to group elements</p>
</div>

<!-- This is the root closing tag. If an element has children,
it needs a closing tag. -->
</html>

```

- elements can be modified and marked for styling later with attributes
- each element has specific attributes catered for them, but all HTML elements can make use of global attributes. All attributes take the form `name="value"`

```

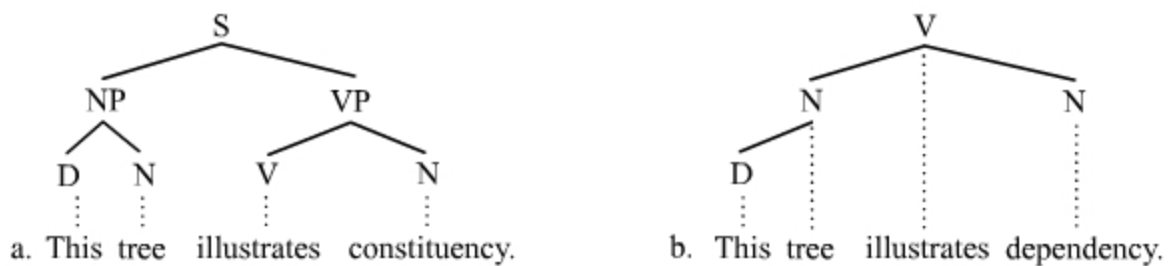
<!-- Two commonly used global attributes are; -->

<!-- 'class' allows a dev to define groups for styling elements later -->
<p class="custom-class-name">This content can be customised later in CSS</p>

<!-- 'id' gives an element a unique ID, mostly for algorithmic reasons -->
<p id="main-page-paragraph-one">Only use an 'id' if you know you need it</p>

```

- the **document flow** is the order in which elements appear on the page
 - it is defined by the parsing of files, i.e breaking down the syntax into basic operations



S = Sentence, NP = Noun Phrase, VP = Verb Phrase, N = Noun, V = Verb, D = Determiner

- by default elements appear in the document flow in the order in which they are written

- this can be modified later but best practice is to write files to reflect the structure of the page you want to build as much as possible

HTML Best Practices

- though HTML is very forgiving and will display a webpage with many errors in the code, best practice is to run it through a validator and remove all errors and warnings
- over the years certain standards have developed. Adherence to them not only improves the quality, clarity and portability of your code, but it caters for Search Engine Optimisation and will earn you a nod of respect from many devs

```
<!DOCTYPE html>

<!-- White space can be added to improve readability -->

<!-- Setting a language is just good manners -->
<html lang="en-gb">

<!-- The <head> holds information that will not appear on the page -->
<head>

    <!-- Tell your browser how to convert human-code into machine-code -->
    <meta charset="utf-8">

    <!-- Always provide a title to label the tab and aid SEO -->
    <title>Infamous Creepers</title>

    <!-- Style with linked files, not inline. Link 'normalize' for portability -->
    <link href="/normalize.css" type="text/css" rel="stylesheet">

    <!-- Link styling files in order of specificity to avoid overriding settings -->
    <link href="/site-specific-styles.css" type="text/css" rel="stylesheet">

    <!-- Use relative file paths wherever possible -->
    <link href="/page-specific-styles.css" type="text/css" rel="stylesheet">
</head>

<!-- The <body> contains the structure and content of the entire page -->
<body>

    <!-- A <header> contains intro content and / or <nav> aids -->
    <header class="creepers-header">

        <!-- A <nav> contains navigational links around the page and site -->
```

```

<nav class="creepers-nav">
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Creepers</a>
  <a href="#">Praise The Spaghetti Monster</a>
</nav>

<!-- The single <h1> will most likely reflect the <title> in the head -->
<h1 class="creepers-title">Infamous Slugs, Spiders and Snakes</h1>

<!-- A <details> element creates a widget containing hidden information -->
<details class="creepers-details">

  <!-- <summary> element is required and used as the label for the widget -->
  <summary class="creepers-summary">Details</summary>
  Only Creepers wanted by the International Criminal Court are included.
</details>
</header>

<!-- The <main> represents the dominant content of the <body> -->
<main class="creepers-main">

  <!-- A <section> is a generic element for large sections, used when there is
  no specific element to represent it. It should always have a heading -->
  <section class="slugs-section">
    <h2 class="slugs-heading">Infamous Slugs</h2>

    <!-- An <article> represents a self contained unit of content, e.g a post,
    an article, a blog entry, product card, comment, widget or gadget -->
    <article class="slugs-gargor">
      <h3 class="slugs-name">Gargor the Wide</h3>
      <p class="slugs-info">Known throughout the world as a vicious predator,
      Gargor the Wide is wanted by almost every nation on earth.</p>

      <!-- An <aside> contains information indirectly related to content -->
      <aside class="slugs-reward">
        <p class="slugs-info">$5,000,000 dead or alive. Preferably dead.</p>
      </aside>

      <!-- <figure> elements hold small units of self contained content -->
      <figure class="slugs-picture">

        <!-- Provide 'alt' for media content to assist visually impaired -->
        

        <!-- <figure> elements have an optional <figcaption> element -->
        <figcaption class="slugs-caption">Gargor at war</figcaption>
      </figure>
    </article>

    <!-- Add space to separate logical blocks of code -->
    <article class="slugs-blonzo">
      <h3 class="slugs-name"><mark>Blonzo</mark> the Untimely</h3>
      <p class="slugs-info">The only creature empirically proven to understand

```

```

    non-linear time, <mark>Blonzo</mark> the Untimely has lost control of his
    temporal reference frame, appearing only in those worst possible moments.
  </p>
  <aside class="slugs-reward">
    <p class="slugs-info">$500,000,000 alive ONLY. Execution if harmed</p>
  </aside>
  <figure class="slugs-picture">
    
    <figcaption class="slugs-captions">Bonzo still unseen</figcaption>
  </figure>
</article>

<article class="bounty-board">
  <h2>Critical Information for Success</h2>
  <article class="bounty-card">

    <!-- <mark> important text for SEO reference -->
    <h3 class="bounty-title"><mark>Blonzo</mark> the Untimely</h3>

    <!-- <time> elements improve SEO and allow for reminders etc -->
    <p class="bounty-info"><mark>Blonzo</mark> was last seen staging a coup
      in France in <time datetime="1802-10-23">1802</time> but is rumoured
      to have failed. The head of the CIA, director of intelligence at the
      NSA, key members of GCHQ and MI6 all agree that Blonzo's next move
      must come at precisely <time datetime="2026-10-23T20:37:54">20:37:54
      </time>, ignoring the <time datetime="PT24H0M0S">24h 0m 0s</time>
      error margin to account for the lack of geo-location of course </p>
  </article><!-- .bounty-card -->
</article><!-- .bounty-board -->

<!-- If it is not clear what a closing tag represents, best to comment -->
</section> <!-- .slugs-section -->

<section class="spiders-section">
  ...
  ...
  ...
</section> <!-- .spiders-section -->

<section class="snakes-section">
  ...
  ...
  ...
</section> <!-- .snakes-section -->

<!-- A <footer> contains 'other' info for nearest relevant element -->
<footer class="creepers-footer">
  <p class="harsh-truth">Creepers of all kinds will have died during the
    reading of this webpage</p>
</footer>
</main> <!-- .creepers-main -->
<footer class="website-footer">
  <nav class="website-nav">
    <a href="#">Home</a>

```

```

    <a href="#">About</a>
    <a href="#">Creepers</a>
    <a href="#">Praise The Spaghetti Monster</a>
  </nav> <!-- .website-nav -->
</footer> <!-- .website-footer -->
</body>
</html>

```

- more detailed descriptions are available online but be careful, highlander rules apply; there can be only one standard but all conflict, and the original is terrible in almost every way.

Useful Syntax

- **inline elements** alter the appearance of small parts of text, and should be used sparingly

```

<p><em>This text</em> will be emphasised, usually with italics</p>

<p><strong>This text</strong> will be highlighted, usually in bold</p>

<p><span>This text</span> can be styled in any manner of ways</p>

```

- **lists** do what you would expect and can be ordered or unordered

```

<ol>
  <li>This is</li>
  <li>An ordered</li>
  <li>List</li>
</ol>

<ul>
  <li>This is</li>
  <li>An unordered</li>
  <li>List</li>
</ul>

```

- **media** can be embedded as `` `<video>` `<audio>` `<iframe>` or `<embed>` tags

```



<!-- 'controls' will display the media controls on the player -->
<video src="/path/to/video.mp4" controls>

<!-- <audio> elements contain sources, the first supported will be used -->
<audio controls>
  <source src="horse.mp3" type="audio/mpeg">
  <source src="horse.ogg" type="audio/ogg">
  This will display if the browser doesn't support audio tags.
</audio>

<!-- <iframe> elements can embed other pages or documents - include a title -->
<iframe src="https://www.w3schools.com" title="W3Schools"></iframe>

<!--<embed> elements allows general embedding, but should be avoided -->
<embed type="text/html" src="snippet.html" width="500" height="200">

```

- **links** are used to create internal and external "jump-to" points with anchors

```

<!-- A link to an external site, target="_value" determines where to open -->
<a href="http://www.google.com" target="_blank">Google it you muppet!</a>

<!-- A link to an internal file or page -->
<a href="/path/to/local/file.html">Move to another page</a>

<!-- A link to an element with id="unique-id" -->
<a href="#unique-id">Jump around the page</a>

<!-- Other elements can be turned into links -->
<a href="http://www.codecademy.com">
  
</a>

```

- **tables** are used for structuring data into rows and columns

```

<table>
  <!-- The head contains column headers -->
  <thead>
    <tr>
      <th>heading 1</th>
      <th>heading 2</th>
    </tr>
  </thead>

  <!-- The body contains the data -->
  <tbody>

```

```

        <tr>
            <td colspan="2">data over 2 columns</td>
        </tr>
    </tbody>

    <!-- The footer contains footer / summary content -->
    <tfoot>
        <tr>
            <td>summary of col 1</td>
            <td>summary of col 2</td>
        </tr>
    </tfoot>
</table>

```

- **forms** are used to wrap, validate and handle user input

```

<!-- Specify where you want data handled with an 'action' -->
<form action="/place-to-process-data.php">

    <!-- Attach labels to inputs by matching for="element-id" -->
    <label for="mundane-comment">How was you day?</label>

    <!-- Data must take a name="value" format, so set names of inputs -->
    <input type="text" name="mundane-comment" id="mundane-comment"
        placeholder="Placeholder text in textbox before user input">

    <!-- 'checkbox's can be grouped together by giving them all the same name -->
    <input type="checkbox" name="breakfast" id="bacon" value="bacon">
    <label for="bacon">Bacon 🥓 </label>
    <input type="checkbox" name="breakfast" id="eggs" value="eggs" >
    <label for="eggs">Eggs 🍳 </label>
    <input type="checkbox" name="breakfast" id="pancakes" value="pancakes" >
    <label for="pancakes">Pancakes 🥞 </label>

    <!-- 'number's can be restricted by min and max values -->
    <label for="h-years">Enter highlander years:</label>
    <input type="number" name="h-years" id="h-years" min="0" max="2000000000"/>

    <!-- A 'range' provides an interactive slider -->
    <label for="unsure">Estimate uncertainty:</label>
    <input type="range" name="unsure" id="unsure" min="0" max="10" step="0.1">

    <!-- 'radio's allow a user to choose between options (XOR) - set same name -->
    <input type="radio" name="fate" id="death" value="death" />
    <label for="death">Death</label>
    <input type="radio" name="fate" id="taxes" value="taxes" />
    <label for="death">Taxes</label>

    <!-- 'password' hides input from the screen but is super easy to hack -->
    <label for="password">Tell me a secret every super-nerd will know:</label>
    <input type="password" name="password" id="password" required/>

```



```

<!-- <select> can be used for dropdowns, data sent is s-name="o-value" pair -->
<label for="universe-option"> Select a universe type:</label>
<select name="universe-option" id="universe-option">
  <option value="fractal">Fractal</option>
  <option value="fracked">Fracked</option>
  <option value="apocalyptic">Apocalyptic</option>
</select>

<!-- A basic search/autocomplete functionality can be achieved by pairing an
<input> with a <datalist>, input-list-value must equal datalist-id-value -->
<label for="basis">Select basis for life:</label>
<input list="basis-list" id="basis">
<datalist id="basis-list">
  <option value="Carbon">
  <option value="Silicon">
  <option value="Germanium">
  <option value="Tin">
  <option value="Lead">
  <option value="Flerovium">
</datalist>

<!-- To take large amount of user text input a 'textarea' is often used -->
<label for="comment">How are you today?</label>
<textarea rows="10" cols="30" id="comment"></textarea>

<!-- 'submit' defines a button to submit data to the 'action' of the <form> -->
<input type="submit" value="Submit Form Data">
</form>

```