HTML Essentials

HTML Overview & HTML Elements

Lesson Objectives

- Overview about HTML
- HTML elements basic
- Understanding about errors and able to debugging webpages
- Able to create a List

Session 1

HTML OVERVIEW & HTML ELEMENTS

Overview

1. What is HTML
2. Anatomy of HTML document
3. Head & Metadata
4. Anatomy of HTML element
5. Element attribute
6. Nesting Element
7. Block & Inline
8. HTML Comments

1. What is HTML?

- ***HTML*** stands for **H**yper **T**ext **M**arkup **L**anguage
  - HTML describes the structure of a Web page
  - HTML consists of a series of elements
  - HTML elements tell the browser how to display the content
  - HTML elements are represented by tags

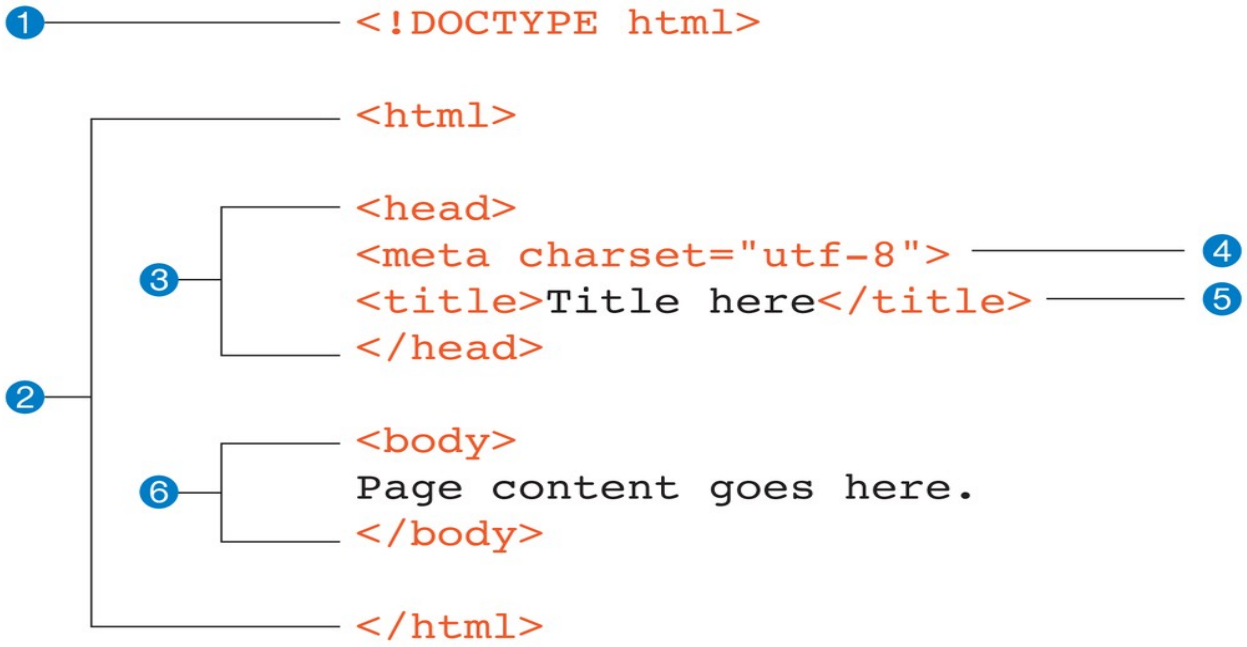
1. What is HTML?

- Example of a basic HTML Document

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image">
  </body>
</html>
```

2. Anatomy of HTML document

- ***Anatomy of HTML document***



3. Head & metadata

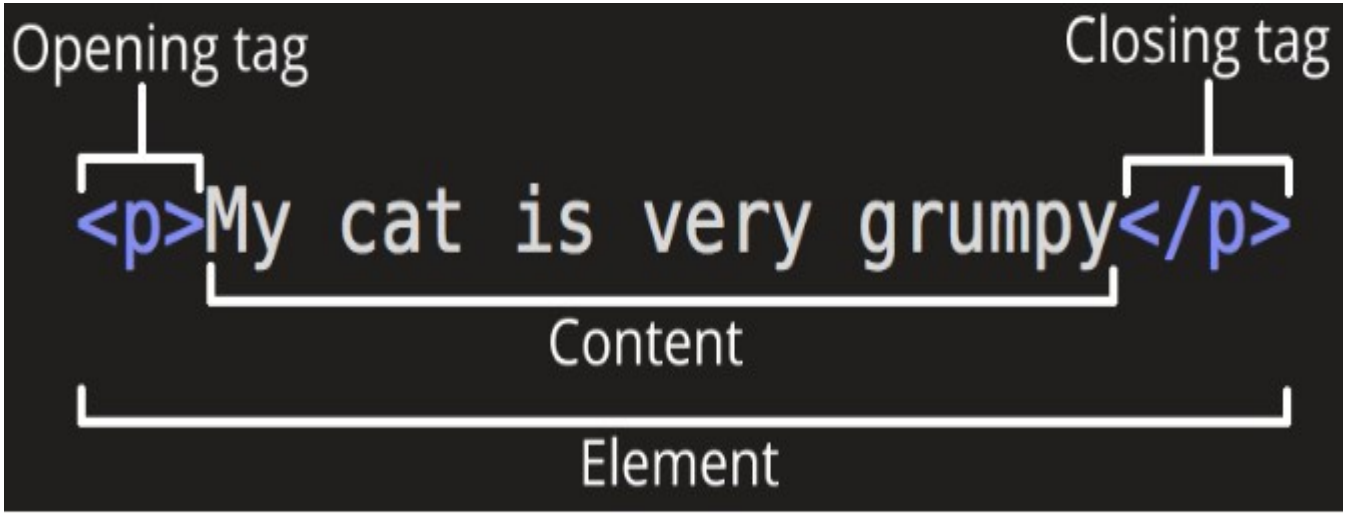
- ***Head*** is a container for all the stuff you want to include on the HTML page that *isn't* the content you are showing to your page's viewers
  - Meta
  - Link
  - Title
  - Style
  - Script

3. Head & metadata

- ***Metadata*** is data about data. The **meta** tag (<meta>) is used in an HTML document to provide high level metadata about the web page, such as:
  - A page description and the keywords that describe the subject of the page.
  - Page authorship information.
  - Instructions for specific browser actions.
  - Details about the page title, description, and author to be used when the page is posted on social media or shown in SERPs.

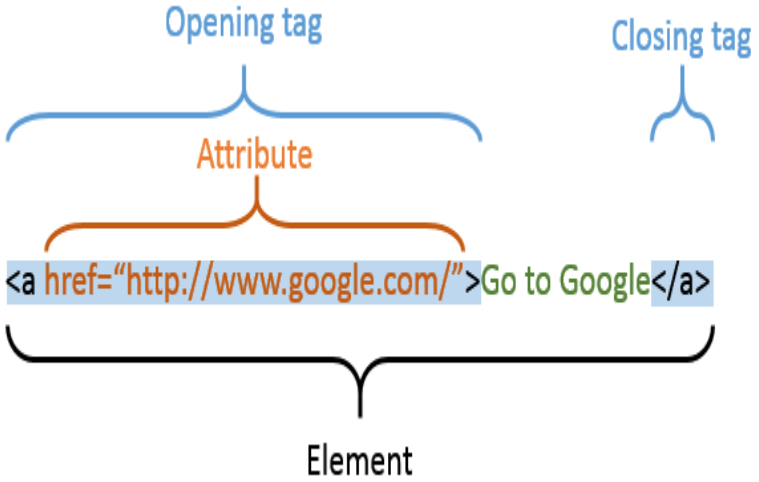
4. Anatomy of HTML Element

- ***Anatomy of HTML Element***



5. Element attribute

- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**



- ***Id attribute:*** Specifies a unique id for an HTML element
- **Class attribute:** Define equal styles for elements with the same class name.

6. Nesting elements

- ***Nesting elements***
  - HTML elements can be nested (elements can contain elements).
  - All HTML documents consist of nested HTML elements

❖ ***Note:*** Make sure that your elements are properly nested

```
1 <p>My cat is <strong>very</strong> grumpy.</p>
```

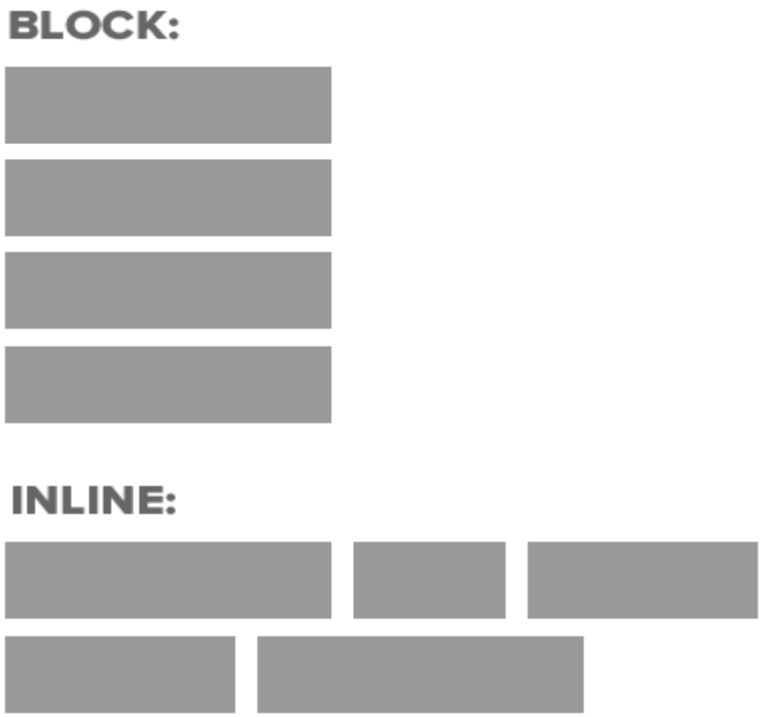
=> Correct

```
1 <p>My cat is <strong>very grumpy.</p></strong>
```

=> Incorrect

7. Block & Inline

Take a look



7. Block & Inline

- ***Block elements***:
  - Always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).
  - Block elements in HTML

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>	
<dl>	<dt>	<fieldset>	<figcaption>	<figure>	<footer>	<form>
<h1>-<h6>	<header>	<hr>		<main>	<nav>	<noscript>
	<p>	<pre>	<section>	<table>	<tfoot>	
<video>						

7. Block & Inline

- ***Inline elements***:
  - Does not start on a new line and only takes up as much width as necessary.
  - Inline elements in HTML

<a>	<abbr>	<acronym>	<b>	<bdo>	<big>	
<button>	<cite>	<code>	<dfn>		<i>	<img>
<input>	<kbd>	<label>	<map>	<object>	<output>	<q>
<samp>	<script>	<select>	<small>			<sub>
<sup>	<textarea>	<time>	<tt>	<var>		

8. HMTL Comments

- ***HTML comments***:
  - Comment tags are used to insert comments in the HTML source code
  - An HTML comment begins with <!– – and the comment closes with – –>

- You can add comments to your HTML source by using the following syntax:

```
<!-- Write your comments here -->
```

Session 2

DEBUGGING HTML

Overview

1. Syntax error
2. Logic error
3. Dom inspector
4. HTML Validator

1. Syntax error and Logic error

- There are two main types of error that you'll come across:
  - **Syntax error**: Relate to spelling errors in your code.

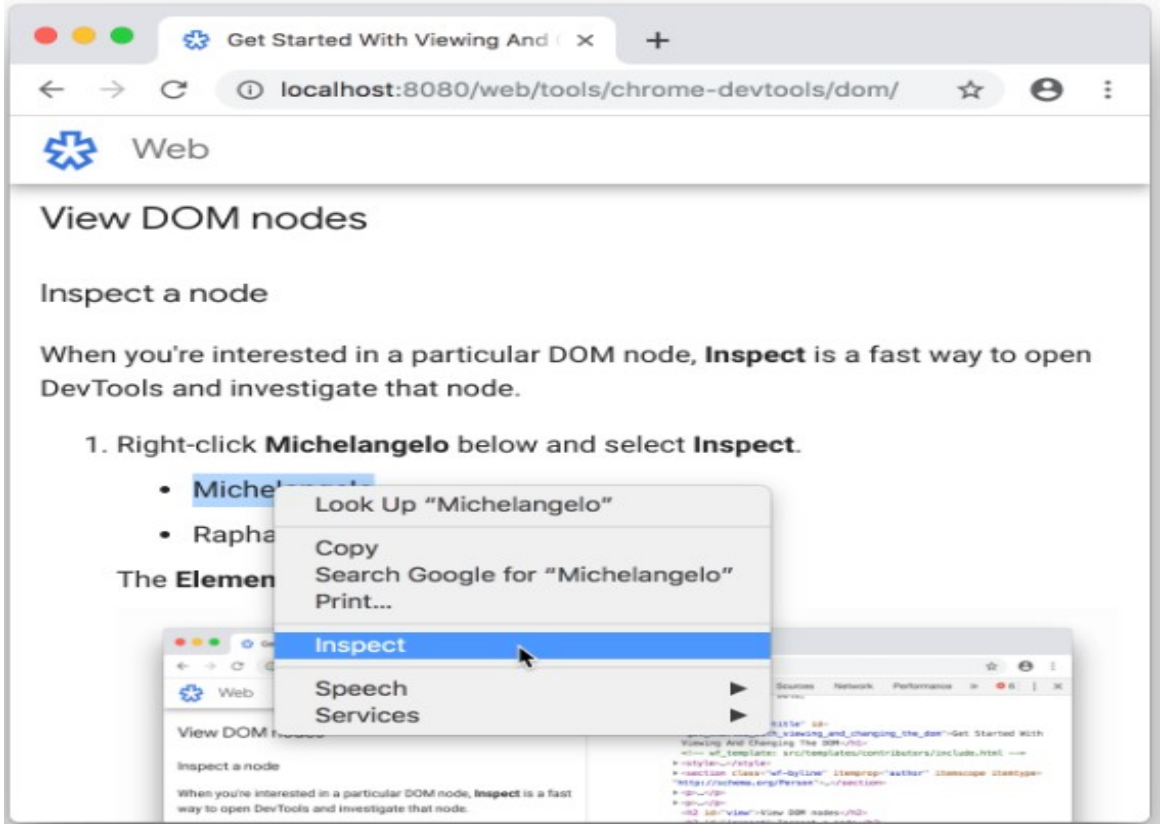
  => Easy to fix as long as you are familiar with the language's syntax and know what the error messages mean.
  - **Logic error:** The syntax is actually correct, but the code is not what you intended it to be so the program runs incorrectly

  => Often harder to fix than syntax errors, because there isn't an error message to direct you to the source of the error.

2. Dom inspector

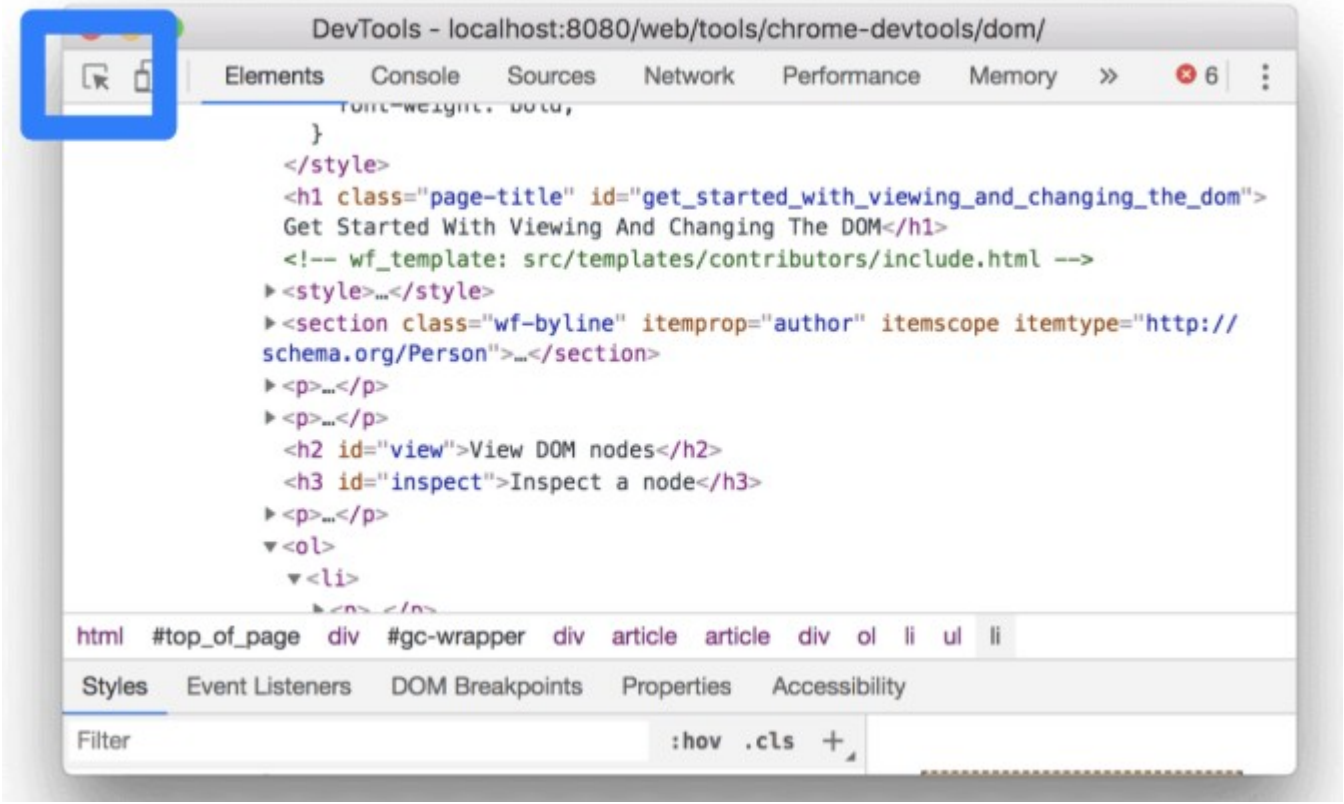
- **Dom Inspector:** A developer tool used to inspect, browser, and edit the Document Object Model of documents - usually web pages
- Basic actions of the DOM Nodes viewer:
  - ✓ Selecting elements by click
  - ✓ Searching for Nodes in the DOM
  - ✓ Updating the DOM Dynamically

2. Dom inspector



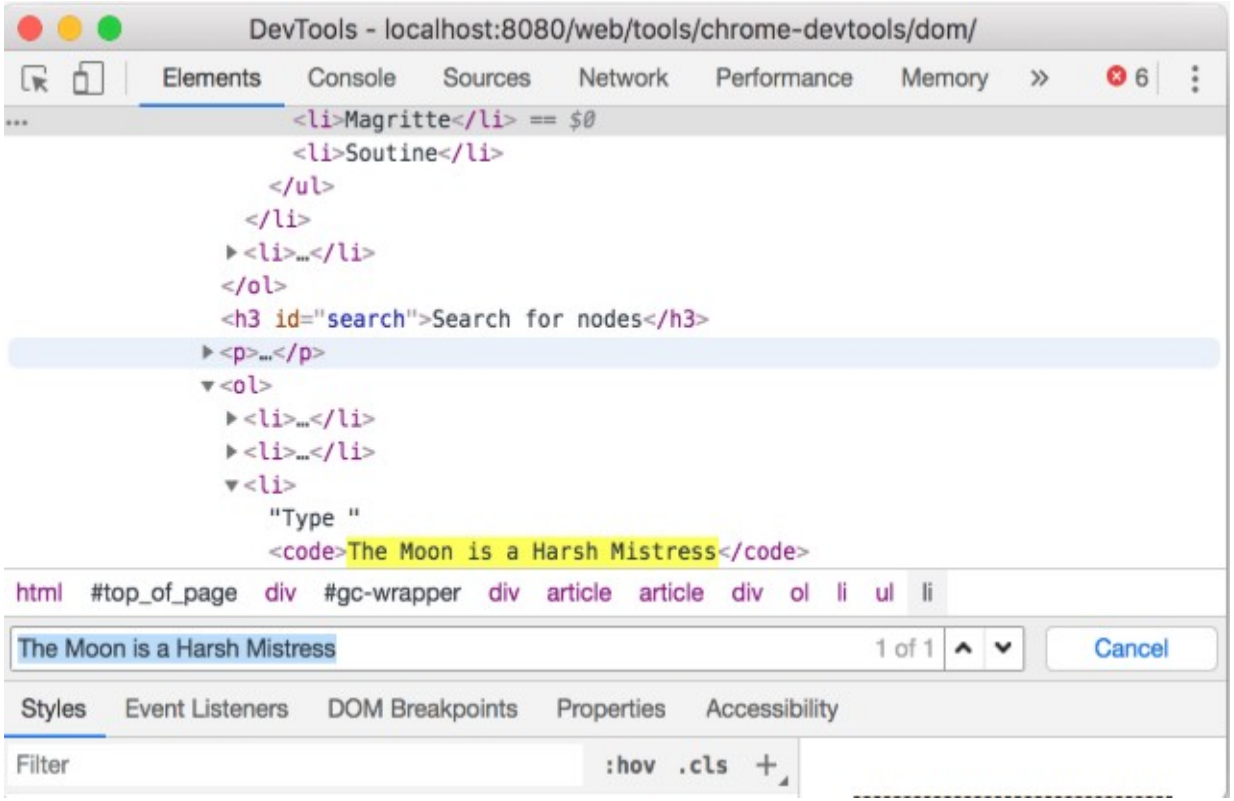
=> Right click on the web page to use Dom Inspector

2. Dom inspector



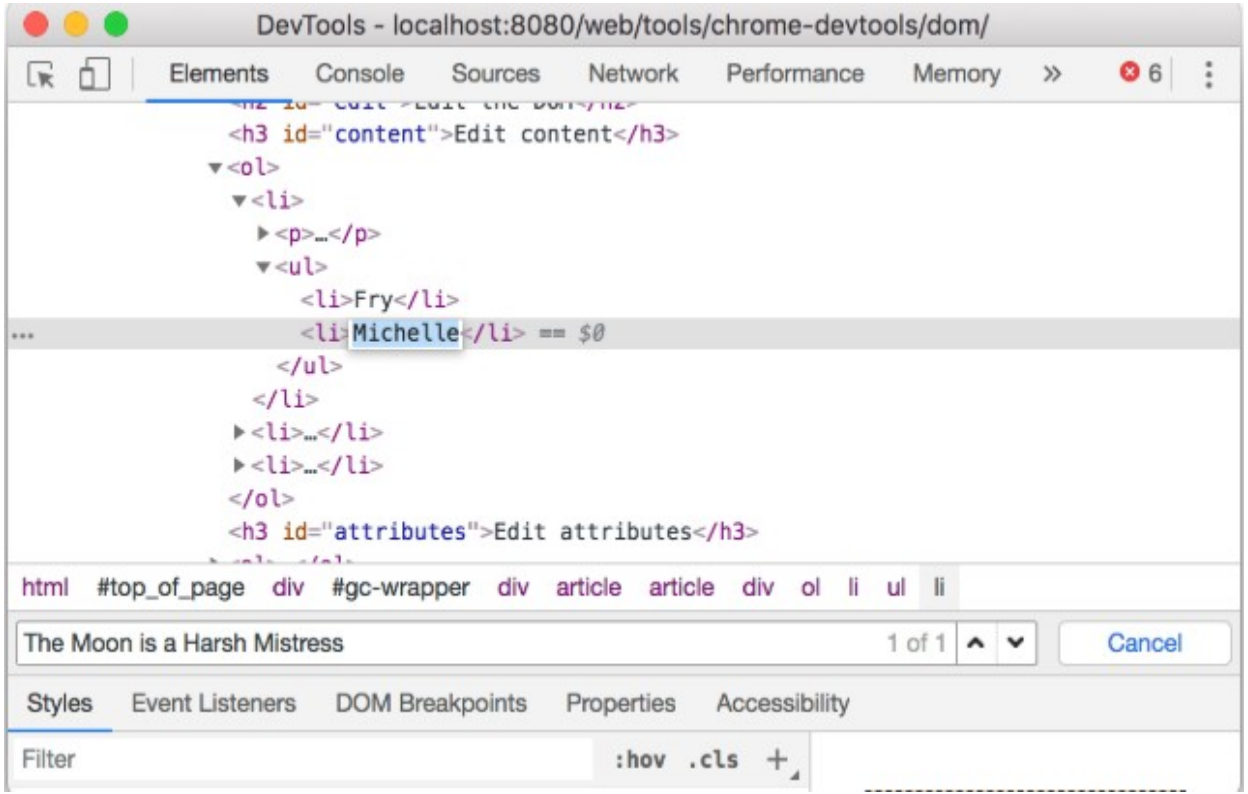
=> Click to choose the node

2. Dom inspector



=> Press Control+F, Type to the search bar at the bottom of the DOM Tree

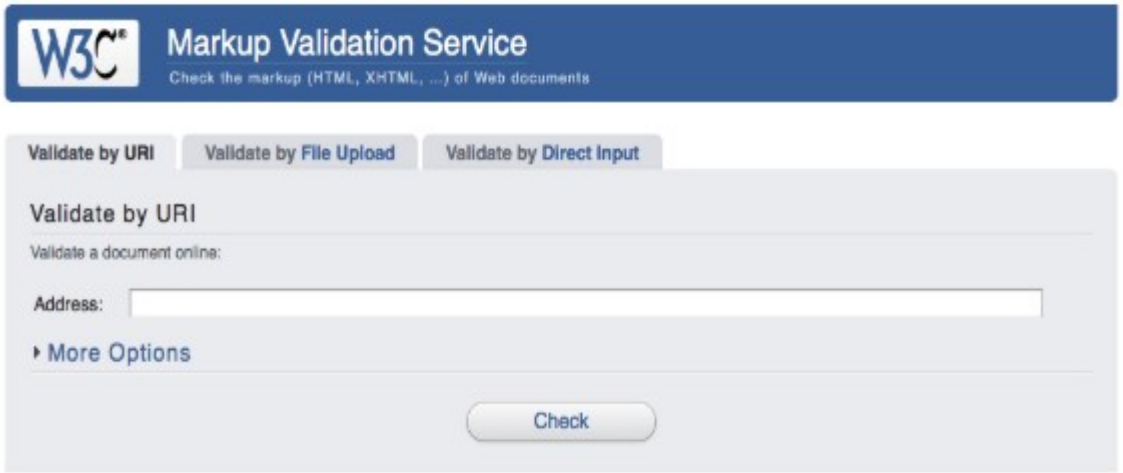
2. Dom inspector



=> Double click to the content or the tag to edit then press Enter to confirm the change

3. HTML Validator

- **HTML Validator:** We can check error for our HTML Document by running your HTML page through the Markup Validation Service via https://validator.w3.org/



Session 3

HTML TEXT

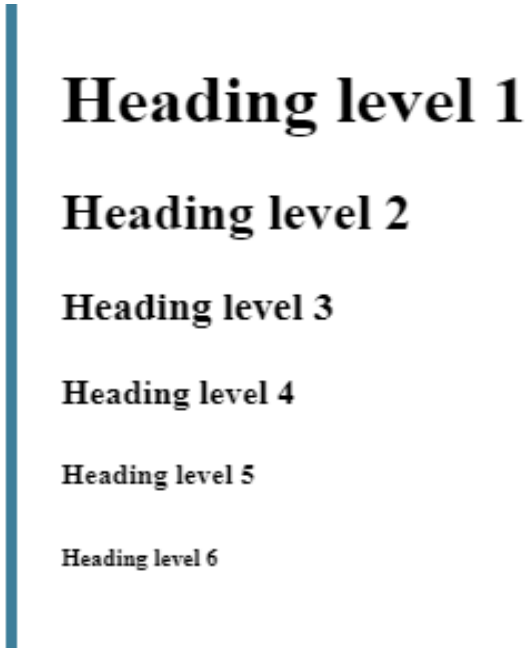
Overview

1. Heading
2. Paragraph
3. Structural hierarchy
4. Why struture is important?
5. Semantics
6. Lists
7. Nesting list
8. Emphasis & Important
9. Italic, bold, underline

1. Heading

- **Heading:** There are six heading elements — <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>.

```
1 <h1>Heading level 1</h1>
2 <h2>Heading level 2</h2>
3 <h3>Heading level 3</h3>
4 <h4>Heading level 4</h4>
5 <h5>Heading level 5</h5>
6 <h6>Heading level 6</h6>
```



1. Heading

- **Heading:** Each element represents a different level of content in the document; <h1> represents the main heading, <h2> represents subheadings, <h3> represents sub-subheadings, and so on.

- **Purpose of Heading:**
  - Search engines use the headings to index the structure and content of your web pages.
  - Users often skim a page by its headings. It is important to use headings to show the document structure.

Note: Use HTML headings for headings only. Don't use headings to make text **BIG** or **bold**.

1. Heading

- **Usage notes:**
  - Avoid using heading tags to resize text. Instead, use the CSS font-size property.
  - Avoid skipping heading levels: always start from <h1>, next use <h2> and so on.
  - You should only use one <h1> per page.

2. Paragraph

- **Paragraph:** <p> element represents a paragraph. HTML paragraphs can be any structural grouping of related content, such as images or form fields.

```
<p>This is the first paragraph of text.
  This is the first paragraph of text.
  This is the first paragraph of text.
  This is the first paragraph of text.</p>
<p>This is the second paragraph.
  This is the second paragraph.
  This is the second paragraph.
  This is the second paragraph.</p>
```

This is the first paragraph of text. This is the first paragraph of text. This is the first paragraph of text. This is the first paragraph of text.

This is the second paragraph. This is the second paragraph. This is the second paragraph. This is the second paragraph.

3. Structure hierarchy

- **Structure hierarchy:**
  - Should just use a single <h1> per page — this is the top level heading, and all others sit below this in the hierarchy.
  - Make sure you use the headings in the correct order in the hierarchy.
  - Of the six heading levels available, you should aim to use no more than three per page, unless you feel it is necessary.

4.Why structure hierarchy is important?

- **Why structure hierarchy is important?**
  - Users looking at a web page tend to scan quickly to find relevant content, often just reading the headings to begin.
  - Search engines indexing your page consider the contents of headings as important keywords for influencing the page's search rankings.
  - To style content with CSS, or make it do interesting things with Javascript, you need to have elements wrapping the relevant content, so CSS/JavaScript can effectively target it.

5. Semantics

- **Semantics:** A semantic element clearly describes its meaning to both the browser and the developer.
  - Examples of **non-semantic** elements:

    `<div>` and `<span>` - Tells nothing about its content.

  - Examples of **semantic** elements:

    `<form>`, `<table>`, and `<article>` - Clearly defines its content.

5. Semantics

- **Compare Semantics and Non-Semantics**

```
<header></header>
<section>
    <article>
        <figure>
            <img>
            <figcaption></figcaption>
        </figure>
    </article>
</section>
<footer></footer>
```

VS

```
<div id="header"></div>
<div class="section">
    <div class="article">
        <div class="figure">
            <img>
            <div class="figcaption"></div>
        </div>
    </div>
</div>
<div id="footer"></div>
```

5. Semantics

- **Compare Semantics and Non-Semantics**
  - The semantics is much **easier to read.**
  - The semantics has **greater accessibility**: Well-support for Search Engine.
  - The semantics lead to more **consistent code**.

5. Semantics

- **Some semantics elements in HTML5**

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

6. List

- **HTML Lists:** Are used to present list of information in well formed and semantic way.
- There are three different types of list in HTML and each one has a specific purpose and meaning.
  - **Unordered list** — Used to create a list of related items, in no particular order.
  - **Ordered list** — Used to create a list of related items, in a specific order.
  - **Description list** — Used to create a list of terms and their descriptions.

6. List

➢ Unorder list

Example — Try this code »

```
<ul>
    <li>Chocolate Cake</li>
    <li>Black Forest Cake</li>
    <li>Pineapple Cake</li>
</ul>
```

=> output

- Chocolate Cake
- Black Forest Cake
- Pineapple Cake

6. List

- **Order list**

Example — Try this code »

```
1  <ol>
2      <li>Fasten your seatbelt</li>
3      <li>Starts the car's engine</li>
4      <li>Look around and go</li>
5  </ol>
```

=> output

1. Fasten your seatbelt
2. Starts the car's engine
3. Look around and go

6. List

- **Description list**

Example — Try this code »

```
<dl>
    <dt>Bread</dt>
    <dd>A baked food made of flour.</dd>
    <dt>Coffee</dt>
    <dd>A drink made from roasted coffee beans.</dd>
</dl>
```

=> output

Bread
    A baked food made of flour.
Coffee
    A drink made from roasted coffee beans.

7. Nesting list

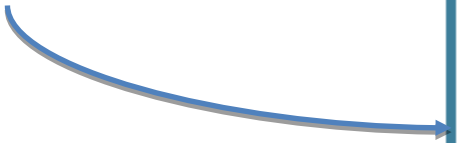
- **Nesting list:** List can be nested (lists inside lists):

```
<ol>
  <li>Remove the skin from the garlic, and chop coarsely.</li>
  <li>Remove all the seeds and stalk from the pepper, and chop coarsely.</li>
  <li>Add all the ingredients into a food processor.</li>
  <li>Process all the ingredients into a paste.
    <ul>
      <li>If you want a coarse "chunky" hummus, process it for a short time.</li>
      <li>If you want a smooth hummus, process it for a longer time.</li>
    </ul>
  </li>
</ol>
```

8. Emphasis & Important

- **Emphasis:** The HTML element defines emphasized text, with added semantic importance.

```
<p>
  In HTML 5, what was previously called
  <em>block-level</em> content is now called
  <em>flow</em> content.
</p>
```



In HTML 5, what was previously called *block-level* content is now called *flow* content.

8. Emphasis & Important

- **Strong Important:** The HTML element defines strong text, with added semantic "strong" importance.

```
1 | <p>Before proceeding, <strong>make sure you put on your safety goggles</strong>.</p>
```

Before proceeding, **make sure you put on your safety goggles**.

9. Italic, bold, underline

Italic, bold, underline: Are formatting elements were designed to display special types of text.

- **<i>** is used to convey a meaning traditionally conveyed by italic: Foreign words, taxonomic designation, technical terms, a thought...
- **<b>** is used to convey a meaning traditionally conveyed by bold: Key words, product names, lead sentence...
- **<u>** is used to convey a meaning traditionally conveyed by underline: Proper name, misspelling..

Thank you