WESTCLIFF UNIVERSITY
Educate. Inspire. Empower.

Week 1 Day 1: HTML Foundation

Agenda

- Coding Software Installation
- Web Service Fundamentals
- Front end vs. Back end vs. Full stack
- Basics of HTML, CSS, and JavaScript and how they are used
- HTML building blocks
- Inline vs. Block elements
- Classes Vs ID
- Common Content Markup Elements
- Semantic HTML
- HTML comments
- Resources
- In-Class Exercise/Demo
- Homework

Coding Software Installation

Coding Software

- Throughout this Bootcamp, we will be using **Visual Studio Code** to code. If you are not familiar with **VS Code**, visit this link to get more information and also to download and install on your computer:

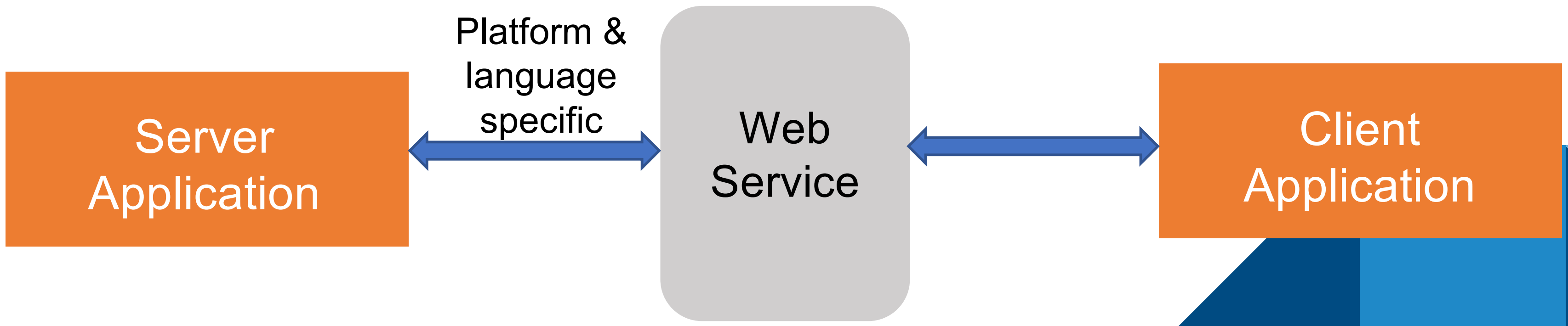
  https://code.visualstudio.com

- After installing, learn how to get started by heading to:
  - https://code.visualstudio.com/docs
- Also recommended are installing extensions to enhance your experience when using **VS Code**.
  - Head to: https://marketplace.visualstudio.com/VSCode. Suggested extension to start: ***prettier***.

Web Service Fundamentals

Web Service Fundamentals

- Web services include any software, application, or cloud technology that provides standardized web protocols (HTTP or HTTPS) to interoperate, communicate, and exchange data messaging.
- A key feature of web services is that applications can be written in various languages and are still able to communicate by exchanging data with one another via a web service between clients and servers.



Web Service Fundamentals

- What are the Different Types of Web Services?
  - XML-RPC, UDDI, SOAP and REST
- API vs. Web Services
  - Most web services provide an API, which, with its set of commands and functions, is used to retrieve data. Example: Twitter delivers an API that authorizes a developer to access tweets from a server and then collects data in JSON format.
  - Something to keep in mind: All web services can be APIs, but not all APIs can be web services.

Frontend vs. Backend vs. Fullstack

Frontend vs. Backend vs. Fullstack

- Front End
  - Front end refers to the front view of a website that users see and interact with, from fonts, colors to dropdown menus and sliders. They are the result of HTML, CSS, and JavaScript at play together that are translated to visuals by web browsers on the computer.
- Back End
  - Back end refers to technologies that host, retrieve, update and remove contents of an application. Back end technologies typically consist of a server, an application, and a database. A back-end developer builds and maintains the technology that powers those components which, together, enable the user-facing side of the website.

Frontend vs. Backend vs. Fullstack

- Full Stack
    - Popularized by Facebook’s engineering department, the idea is that a full stack developer can work cross-functionally on the full “stack” of technology, i.e. both the front end and back end. Full stack developers offer the full package.

Basics of HTML, CSS, and JavaScript and how they are used

Basics of HTML, CSS, and JavaScript and how they are used

An overview:

- **HTML** provides the *basic structure* of sites (known as structural or markup language), which is enhanced and modified by other technologies like CSS and JavaScript.
- **CSS** is used to control *presentation, formatting, and layout* – hence known as presentation language.
- **JavaScript** is used to control the *behavior* of different elements and is known as behavioral language.

Basics of HTML, CSS, and JavaScript and how they are used

How they are used:

- **HTML** stands for Hyper Text Markup Language. "Markup language" means that, rather than using a programming language to perform functions, HTML uses tags to identify different types of content and the purposes they each serve to the webpage.
- Current version: HTML5
- Example of an HTML tag: <body>
- Most HTML tags have a pair of open and close tags. Example: <h1></h1>
- Contents are wrapped within them. Example: <h1>Today’s Headline</h1>
- The text content: Today’s Headline is therefore markup using HTML tags.

Basics of HTML, CSS, and JavaScript and how they are used

How they are used:

- **CSS** stands for Cascading Style Sheets. This programming language dictates how the HTML elements (tags) of a website should actually appear visually on the front end of the page.
- Current version: CSS3
- CSS helps style content that are markup with HTML tags so they appear to the user the way it was intended to be seen. These languages are kept separate to ensure websites are built correctly before they're reformatted.
- Example of a CSS code: style1 {color: #c2dde6}

Basics of HTML, CSS, and JavaScript and how they are used

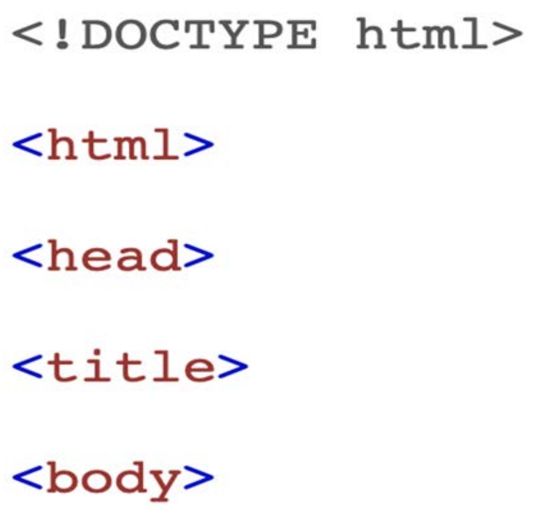
How they are used:

- **JavaScript** is a logic-based programming language that can be used to modify website content and make it behave in different ways in response to a user's actions. Some common uses for JavaScript include confirmation boxes, calls-to-action, customize page content and adding new identities to existing information.
- Current version: ES6
- Just like HTML and CSS, JavaScript is also separated from the 2 languages so as to make content behave as when an event (example: user action) occur.
- Example of JavaScript: `function updateName() { some scripts }`

HTML building blocks

HTML building blocks

Main HTML structural code blocks

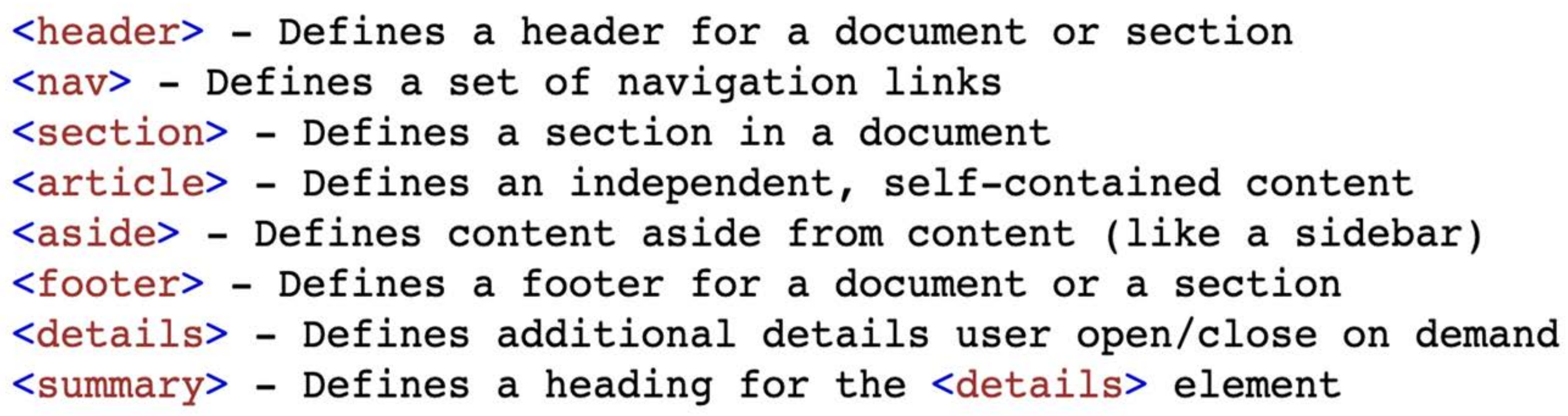


```
<!DOCTYPE html>

<html>

<head>

<title>

<body>
```

HTML building blocks

HTML5 Elements

- These are some of the common HTML5 building block elements

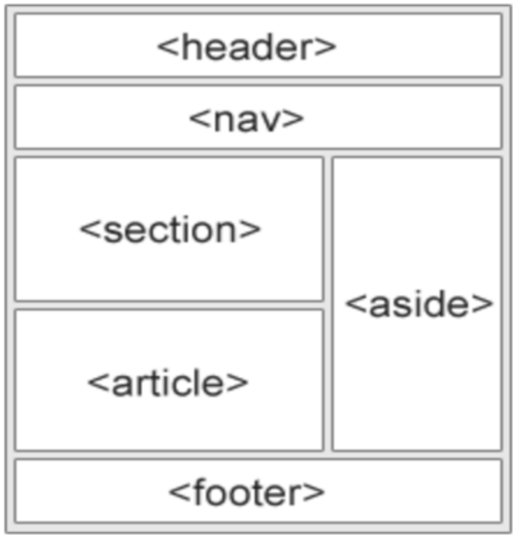


```
<header> - Defines a header for a document or section
<nav> - Defines a set of navigation links
<section> - Defines a section in a document
<article> - Defines an independent, self-contained content
<aside> - Defines content aside from content (like a sidebar)
<footer> - Defines a footer for a document or a section
<details> - Defines additional details user open/close on demand
<summary> - Defines a heading for the <details> element
```

HTML building blocks

HTML5 Elements

- Example of Page Layout using HTML5 elements



HTML building blocks

Simple HTML document markup

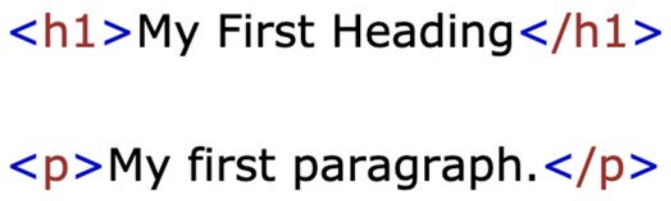
```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

HTML building blocks

HTML Elements

- An HTML element is defined by a start tag, some content, and an end tag.

  Example:

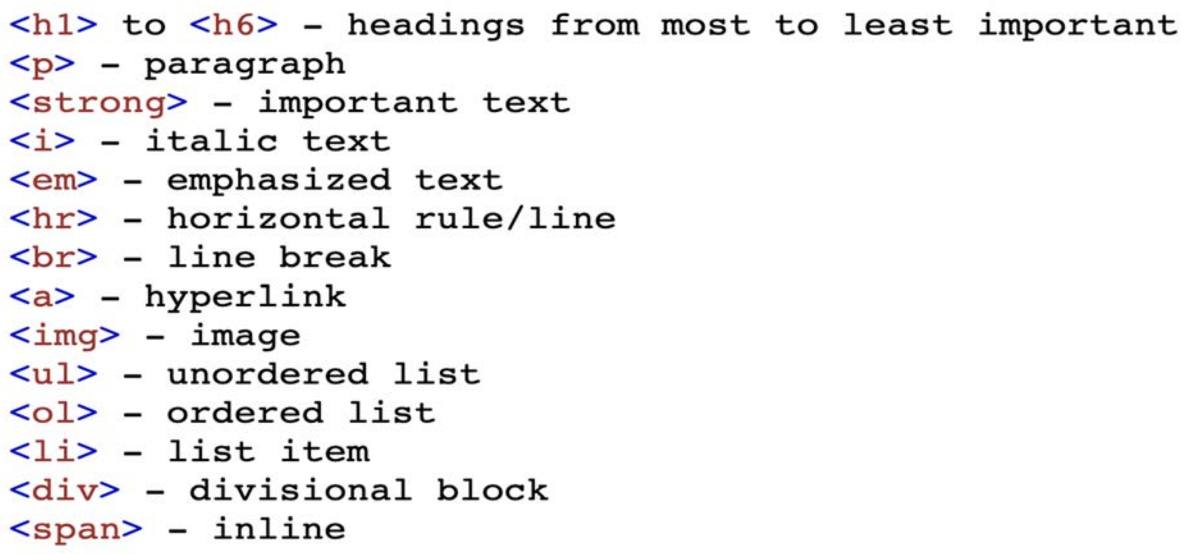


```
<h1>My First Heading</h1>

<p>My first paragraph.</p>
```

HTML building blocks

HTML Elements

- These are some of the most common HTML elements

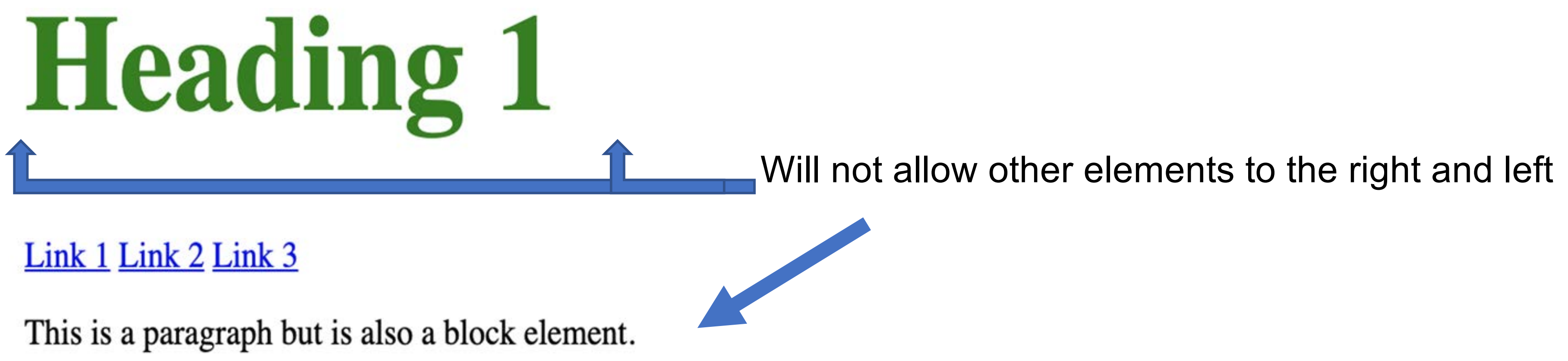


```
<h1> to <h6> - headings from most to least important
<p> - paragraph
<strong> - important text
<i> - italic text
<em> - emphasized text
<hr> - horizontal rule/line
<br> - line break
<a> - hyperlink
<img> - image
<ul> - unordered list
<ol> - ordered list
<li> - list item
<div> - divisional block
<span> - inline
```

Inline vs. Block elements

Inline vs. Block elements

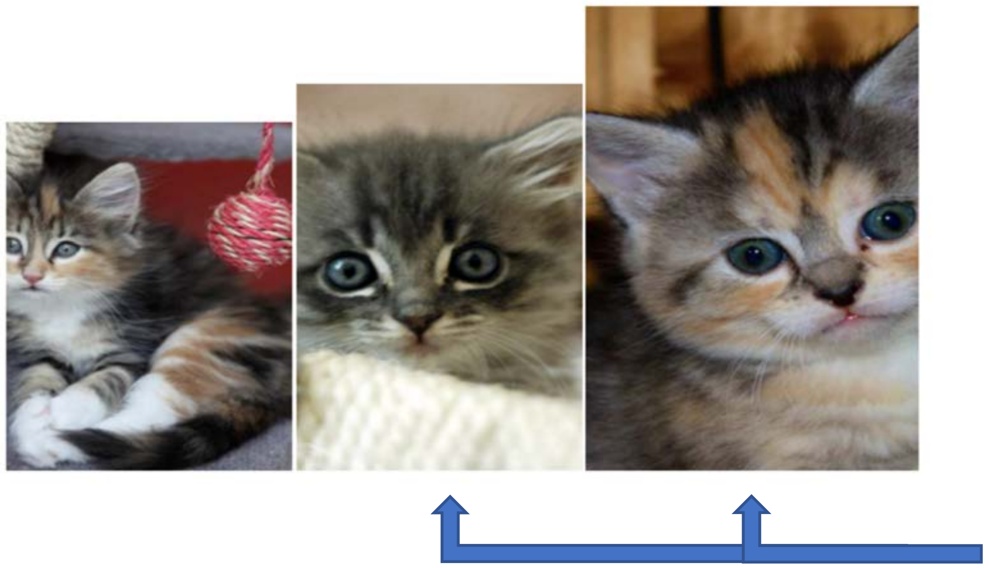
- **Block elements** are those that take up the full width available on a web page, effectively *blocking out* any other elements from sitting next to it on the left or right. Examples of <h1> and <p> elements:



This is a paragraph but is also a block element.

Inline vs. Block elements

- **Inline elements** are those which only take up as much width as is needed to display the contents of the element, thereby allowing other elements to be *in line* with the inline element.

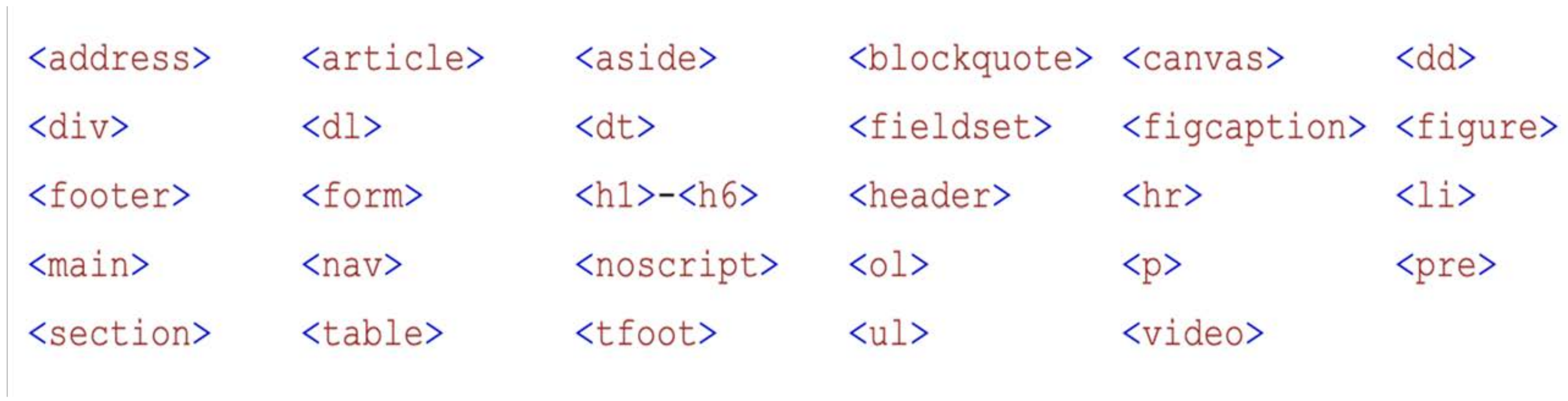


An example of <img> elements.

Allow other elements to the right or left of it

Inline vs. Block elements

- **Block Elements**



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

Inline vs. Block elements

- **Inline Elements**

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

ID Vs Class

ID vs Class

- What is ID and Class?
  - Block and inline elements are often given additional identity so that other languages like CSS and JavaScript can communicate with that element via ID and class.
  - They are essentially just another piece of code added to the element, more specifically the open tag and as an attribute.
- ID and Class Rules
  - ID names are unique. An ID name given on a page cannot be repeated on the same page. They can however be used again on other pages.
  - Unlike ID names, class names can be repeated – on the same page and other pages.
  - When creating ID or class names, do not use symbols other than underscore (_)and dash (-), do not begin names with a number and no spaces between 2 or more words.

ID vs Class

- When to use ID vs Class?
  - Use ID on main block elements ex: body, header, nav, main, footer, div, where their names are unique (not repeated elsewhere on the page).
  - Use class on smaller blocks and inline elements ex: div, ul, li, img, span, where these elements are typically not unique (likely repeat elsewhere on the page).
- How they are used:
  - `<body id="aboutUs">`
  - `<div id="container">`
  - `<img class="gallery1" .........>`
  - `<span class="best_quote">`

Common Content Markup Elements

Common Content Markup Elements

- Block Element: **Divisional**

```
<div>                      Mainly use for grouping contents
        <h2>…..</h2>
        <p>….</p>
        <p>….</p>
</div>
```

- Block Element: **Headings**

```
<h1>….</h1>Most important heading
<h2>…. </h2>
<h3> …. </h3>
<h4> …. </h4>
<h5> …. </h5>
<h6> …. </h6>       Least important heading
```

Common Content Markup Elements

- Block Element: **Paragraphs**

```
<p> …. </p>
```

- Block Element: **List**

Ordered List

```
<ol>
        <li>Item 1</li>
        <li>Item 2</li>
</ol>
```

1. Item 1
2. Item 2

Unordered List

```
<ul>
        <li>Item 1</li>
        <li>Item 2</li>
</ul>
```

- Item 1
- Item 2

Common Content Markup Elements

- Block Element: **Table and its sub-elements**

```
<table>
        <tr>
            <td>Item 1</td>
            <td>Item 2</td>
        </tr>
</table>
```

Item 1	Item 2

- Block Element: **Form** (its sub-elements are however inline not block)

```
<form>
        <label>Your Name: </label>
        <input type="text" name="fullname" placeholder="Full Name">
</form>
```

Your Name: Full Name

Common Content Markup Elements

- Inline Element: **Image**

```
<img src="image.png" alt="Image Name" width="400" height="275">
```

- Inline Element: **Hypertext Link**

```
<a href=”filename.html">
```

- Inline Element: **Line Break**

```
<br>
```

- Inline Element: **Button**

```
<button>….</button>
```

- Inline Element: **Span**

```
<span>….</span>
```

Use for isolating word(s) in a sentence.

Semantic HTML

Semantic HTML

- **Semantic HTML** or semantic markup is the meaning of HTML markups on the web page, not just the presentation. For example, a <p> tag indicates that the enclosed text is a paragraph. This is both semantic and presentational because people know what paragraphs are, and browsers know how to display them.
- On the flip side of this equation, tags such as <b> and <i> are not semantic. They define only how the text should look (bold or italic), and don't provide any additional meaning to the markup.
- For example, use <h1> element to describe a heading not to display a bold sentence. In other words, use the appropriate element for the intended type of content not for the purpose of looks.

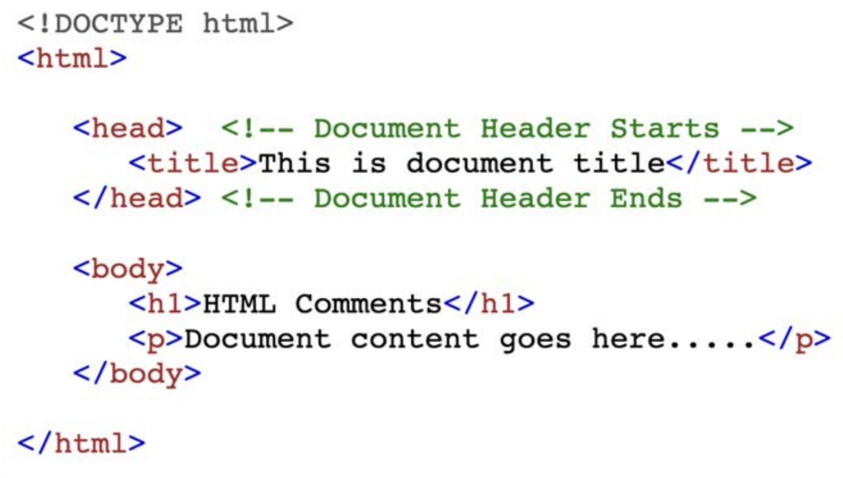
HTML comments

HTML comments

- **HTML comment** is a piece of code that is ignored by web browsers – in other words this comment code will not be displayed on the front end visual but rather on the browser source codes.
- It is good practice to add comments to your HTML document, especially in large projects. This help speeds up in identifying sections of a document and also provide notes to anyone viewing the codes. Comments, therefore, help you and others understand the codes and increases code readability.
- HTML comments are placed in between this tag **<!-- ... -->**. Any content placed within this tag or markers will be treated as comment and will be completely ignored by the browser.
- Example: <!- - Write your comments here - ->

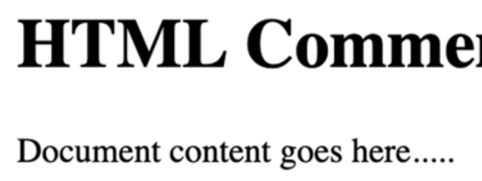
HTML comments

- Here's an example of comments used on an HTML document:



```
<!DOCTYPE html>
<html>

   <head>  <!-- Document Header Starts -->
      <title>This is document title</title>
   </head> <!-- Document Header Ends -->

   <body>
      <h1>HTML Comments</h1>
      <p>Document content goes here.....</p>
   </body>

</html>
```

What it looks like on the browser:

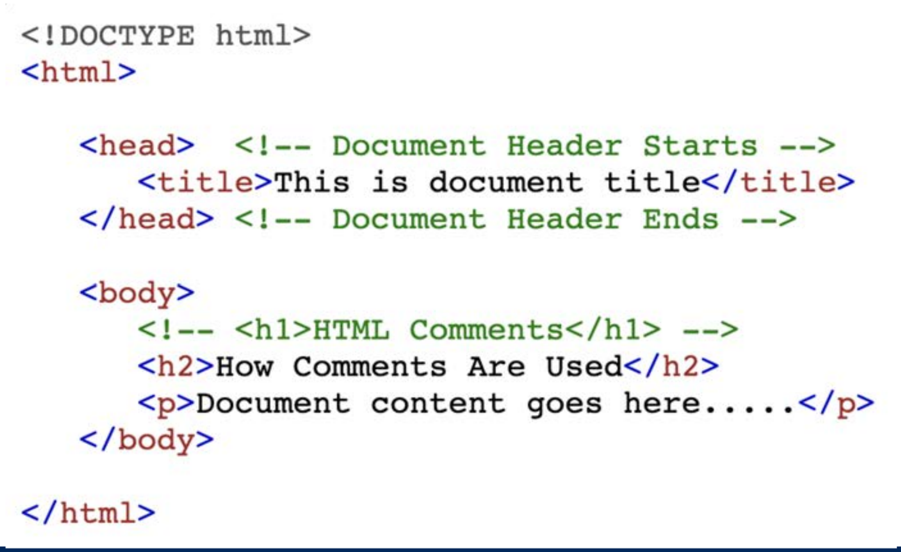


HTML Comments

Document content goes here.....

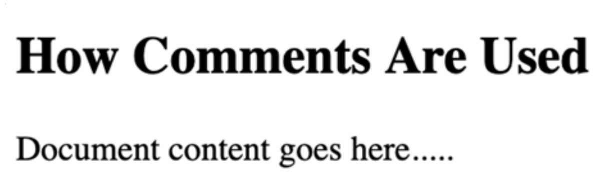
HTML comments

- Another common use of comments is to deactivate codes – instead of erasing unused codes, deactivate it so in the event if you decide to use it, saves you time to rewrite it. Here's an example:



```
<!DOCTYPE html>
<html>

   <head>  <!-- Document Header Starts -->
      <title>This is document title</title>
   </head> <!-- Document Header Ends -->

   <body>
      <!-- <h1>HTML Comments</h1> -->
      <h2>How Comments Are Used</h2>
      <p>Document content goes here.....</p>
   </body>

</html>
```

What it looks like on the browser:



How Comments Are Used

Document content goes here.....

Questions?

Resources

https://www.w3schools.com/html/

https://www.html5rocks.com/en/resources.html

https://www.tutorialspoint.com/html5/index.htm