# Getting Started with the Web

Installing basic Software – VSCODE, notpad++ or any other editor

[What will your website look like?](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/What_will_your_website_look_like)

* Simple Page With Form and Content

[Dealing with files](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/Dealing_with_files)

A website consists of many files: text content, code, stylesheets, media content, and so on. When you're building a website, you need to assemble these files into a sensible structure and make sure they can talk to one another.

[HTML basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)

Hypertext Markup Language (HTML) is the code that you use to structure your web content and give it meaning and purpose. For example, is my content a set of paragraphs or a list of bullet points? Do I have images inserted on my page? Do I have a data table?

[CSS basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics)

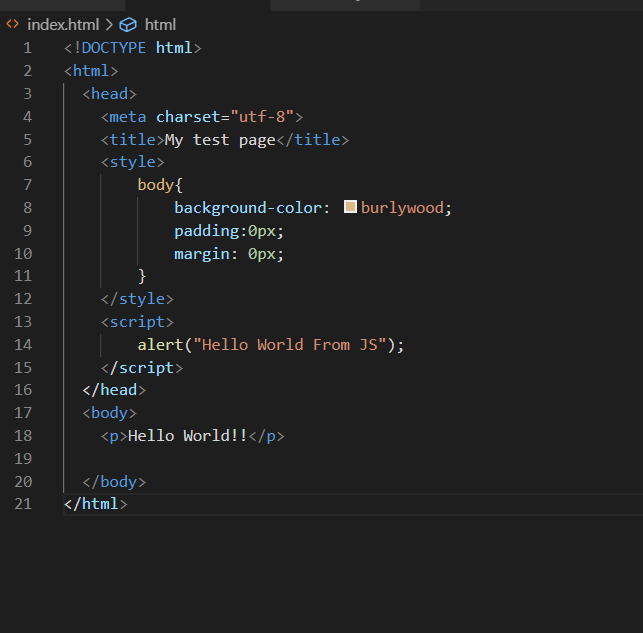
Cascading Stylesheets (CSS) is the code that you use to style your website. For example, do you want the text to be black or red? Where should content be drawn on the screen? What background images and colors should be used to decorate your website?

[JavaScript basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics)

JavaScript is the programming language that you use to add interactive features to your website. Some examples could be games, things that happen when buttons are pressed or data is entered in forms, dynamic styling effects, animation, and much more

**What Is React, Angular, Bootstrap?**

JavaScript frameworks are an essential part of modern front-end web development, providing developers with tried and tested tools for building scalable, interactive web applications. Many modern companies use frameworks as a standard part of their tooling, so many front-end development jobs now require framework experience.



# Getting started with html/html5

## What is HTML?

[HTML](https://developer.mozilla.org/en-US/docs/Glossary/HTML) (Hypertext Markup Language) is not a programming language. It is a markup language that tells web browsers how to structure the web pages you visit. It can be as complicated or as simple as the web developer wants it to be. HTML consists of a series of [elements](https://developer.mozilla.org/en-US/docs/Glossary/Element), which you use to enclose, wrap, or mark up different parts of content to make it appear or act in a certain way. The enclosing [tags](https://developer.mozilla.org/en-US/docs/Glossary/Tag) can make content into a hyperlink to connect to another page, italicize words, and so on.  For example, consider the following line of text:

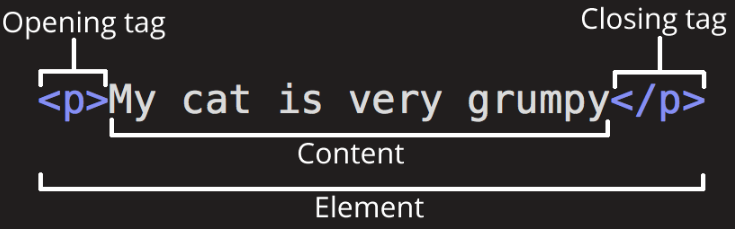
My cat is very grumpy

If we wanted the text to stand by itself, we could specify that it is a paragraph by enclosing it in a paragraph  ([<p>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/p)) element:

<p>My cat is very grumpy</p>

Note: Tags are case-insensitive, However in practice we write in lowercase letter to follow the consistency and readability

## Anatomy of an HTML element



The anatomy of our element is:

* **The opening tag:** This consists of the name of the element (in this example, *p* for paragraph), wrapped in opening and closing angle brackets. This opening tag marks where the element begins or starts to take effect. In this example, it precedes the start of the paragraph text.
* **The content:** This is the content of the element. In this example, it is the paragraph text.
* **The closing tag:** This is the same as the opening tag, except that it includes a forward slash before the element name. This marks where the element ends. Failing to include a closing tag can produce peculiar results.

The element is the opening tag, followed by content, followed by the closing tag.

Nesting elements

Elements can be placed within other elements. This is called *nesting*. If we wanted to state that our cat is **very** grumpy, we could wrap the word *very* in a [<strong>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/strong) element, which means that the word is to have strong(er) text formatting:

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

Block versus inline elements

There are two important categories of elements to know in HTML: block-level elements and inline elements.

* Block-level elements form a visible block on a page. A block-level element appears on a new line following the content that precedes it. Any content that follows a block-level element also appears on a new line. Block-level elements are usually structural elements on the page. For example, a block-level element might represent paragraphs, lists, navigation menus, or footers. A block-level element wouldn't be nested inside an inline element, but it might be nested inside another block-level element.
* Inline elements are contained within block-level elements, and surround only small parts of the document’s content. (not entire paragraphs or groupings of content) An inline element will not cause a new line to appear in the document. It is typically used with text. For example, as an [<a>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a) element (hyperlink) or emphasis elements such as [<em>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/em) or [<strong>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/strong).

Consider the following example

<em>first</em><em>second</em><em>third</em>

<p>fourth</p><p>fifth</p><p>sixth</p>

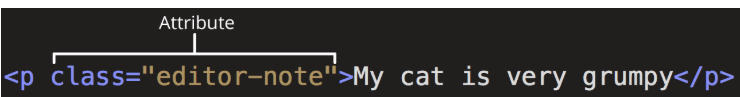
Empty elements

Not all elements follow the pattern of an opening tag, content, and a closing tag. Some elements consist of a single tag, which is typically used to insert/embed something in the document. For example, the [<img>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img) element embeds an image file onto a page:

<img src="https://raw.githubusercontent.com/mdn/beginner-html-site/gh-pages/images/firefox-icon.png">

## Attributes

Elements can also have attributes. Attributes look like this:



Attributes contain extra information about the element that won't appear in the content. In this example, the **class** attribute is an identifying name used to target the element with style information.

An attribute should have:

* A space between it and the element name. (For an element with more than one attribute, the attributes should be separated by spaces too.)
* The attribute name, followed by an equal sign.
* An attribute value, wrapped with opening and closing quote marks.

Another example of an element is <a>. This stands for *anchor*. An anchor can make the text it encloses into a hyperlink. Anchors can take a number of attributes, but several are as follows:

* **href**: This attribute's value specifies the web address for the link. For example: href="https://www.mozilla.org/".
* **title**: The title attribute specifies extra information about the link, such as a description of the page that is being linked to. For example, title="The Mozilla homepage". This appears as a tooltip when a cursor hovers over the element.
* **target**: The target attribute specifies the browsing context used to display the link. For example, target="\_blank" will display the link in a new tab. If you want to display the linked content in the current tab, just omit this attribute.

Boolean attributes

Sometimes you will see attributes written without values. This is entirely acceptable. These are called Boolean attributes. Boolean attributes can only have one value, which is generally the same as the attribute name. For example, consider the [disabled](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#attr-disabled) attribute, which you can assign to form input elements. (You use this to disable the form input elements so the user can't make entries. The disabled elements typically have a grayed-out appearance.) For example:

<input type="text" disabled="disabled">

Single or double quotes?

You might see single quotes in some HTML code. This is a matter of style. You can feel free to choose which one you prefer. Both of these lines are equivalent:

Make sure you don't mix single quotes and double quotes. This example (below) shows a kind of mixing quotes that will go wrong:

<a href="http://www.example.com'>A link to my example.</a>

However, if you use one type of quote, you can include the other type of quote *inside* your attribute values:

<a href="http://www.example.com" title="Isn't this fun?">A link to my example.</a>

## Anatomy of an HTML document

Individual HTML elements aren't very useful on their own. Next, let's examine how individual elements combine to form an entire HTML page:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=500, initial-scale=1">

<title>My test page</title>

</head>

<body>

<p>This is my page</p>

</body>

</html>

Here we have:

1. <!DOCTYPE html>: The doctype. When HTML was young (1991-1992), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML. Doctypes used to look something like this:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

More recently, the doctype is a historical artifact that needs to be included for everything else to work right. <!DOCTYPE html> is the shortest string of characters that counts as a valid doctype. That is all you need to know!

1. <html></html>: The [<html>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/html) element. This element wraps all the content on the page. It is sometimes known as the root element.
2. <head></head>: The [<head>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/head) element. This element acts as a container for eveything you want to include on the HTML page, **that isn't the content** the page will show to viewers. This includes keywords and a page description that would appear in search results, CSS to style content, character set declarations, and more. You'll learn more about this in the next article of the series.
3. <meta charset="utf-8">: This element specifies the character set for your document to UTF-8, which includes most characters from the vast majority of human written languages. With this setting, the page can now handle any textual content it might contain. There is no reason not to set this, and it can help avoid some problems later.
4. <meta name="viewport" content="width=500, initial-scale=1">

width=device-width

The browser's [viewport](https://developer.mozilla.org/en-US/docs/Glossary/viewport) is the area of the window in which web content can be seen.This is often not the same size as the rendered page, in which case the browser provides scrollbars for the user to scroll around and access all the content.

1. <title></title>: The [<title>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/title) element. This sets the title of the page, which is the title that appears in the browser tab the page is loaded in. The page title is also used to describe the page when it is bookmarked.
2. <body></body>: The [<body>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/body) element. This contains *all* the content that displays on the page, including text, images, videos, games, playable audio tracks, or whatever else.

## Entity references: Including special characters in HTML

In HTML, the characters <, >,",' and & are special characters. They are parts of the HTML syntax itself. So how do you include one of these special characters in your text? For example, if you want to use an ampersand or less-than sign, and not have it interpreted as code.

You do this with character references. These are special codes that represent characters, to be used in these exact circumstances. Each character reference starts with an ampersand (&), and ends with a semicolon (;).

| **Literal character** | **Character reference equivalent** |
| --- | --- |
| < | &lt; |
| > | &gt; |
| " | &quot; |
| ' | &apos; |
| & | &amp; |

## HTML comments

To write an HTML comment, wrap it in the special markers <!-- and -->. For example:

<p>I'm not inside a comment</p>

<!-- <p>I am!</p> -->

## Applying CSS and JavaScript to HTML

Just about all websites you'll use in the modern day will employ [CSS](https://developer.mozilla.org/en-US/docs/Glossary/CSS) to make them look cool, and [JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript) to power interactive functionality, such as video players, maps, games, and more. These are most commonly applied to a web page using the [<link>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link) element and the [<script>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script) element, respectively.

* The [<link>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link) element should always go inside the head of your document. This takes two attributes, rel="stylesheet", which indicates that it is the document's stylesheet, and href, which contains the path to the stylesheet file:

<link rel="stylesheet" href="my-css-file.css">

* The [<script>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script) element should also go into the head, and should include a src attribute containing the path to the JavaScript you want to load, and defer, which basically instructs the browser to load the JavaScript at the same time as the page's HTML. This is useful as it makes sure that the HTML is all loaded before the JavaScript runs, so that you don't get errors resulting from JavaScript trying to access an HTML element that doesn't exist on the page yet. There are actually a number of ways to handle loading JavaScript on your page, but this is the most foolproof one to use for modern browsers.

<script src="my-js-file.js" defer></script>

Examples:

Tags – div,p, header, Lists(ul / ol), Link, sup/sub

<a href="mailto:nowhere@mozilla.org">Send email to nowhere</a>

In your HTML code, you can mark up sections of content based on their functionality — you can use elements that represent the sections of content described above unambiguously, and assistive technologies like screenreaders can recognise those elements and help with tasks like "find the main navigation", or "find the main content." As we mentioned earlier in the course, there are a number of [consequences of not using the right element structure and semantics for the right job](https://developer.mozilla.org/en-US/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals#Why_do_we_need_structure).

To implement such semantic mark up, HTML provides dedicated tags that you can use to represent such sections, for example:

**header:**[<header>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/header).

**navigation bar:**[<nav>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nav).

**main content:**[<main>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/main), with various content subsections represented by [<article>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/article), [<section>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/section), and [<div>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div) elements.

**sidebar:**[<aside>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/aside); often placed inside [<main>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/main).

**footer:**[<footer>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/footer).

## HTML layout elements in more detail

It's good to understand the overall meaning of all the HTML sectioning elements in detail — this is something you'll work on gradually as you start to get more experience with web development. You can find a lot of detail by reading our [HTML element reference](https://developer.mozilla.org/en-US/docs/Web/HTML/Element). For now, these are the main definitions that you should try to understand:

* [<main>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/main) is for content unique to this page. Use <main> only once per page, and put it directly inside [<body>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/body). Ideally this shouldn't be nested within other elements.
* [<article>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/article) encloses a block of related content that makes sense on its own without the rest of the page (e.g., a single blog post).
* [<section>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/section) is similar to <article>, but it is more for grouping together a single part of the page that constitutes one single piece of functionality (e.g., a mini map, or a set of article headlines and summaries). It's considered best practice to begin each section with a [heading](https://developer.mozilla.org/en-US/Learn/HTML/Howto/Set_up_a_proper_title_hierarchy); also note that you can break <article>s up into different <section>s, or <section>s up into different <article>s, depending on the context.
* [<aside>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/aside) contains content that is not directly related to the main content but can provide additional information indirectly related to it (glossary entries, author biography, related links, etc.).
* [<header>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/header) represents a group of introductory content. If it is a child of [<body>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/body) it defines the global header of a webpage, but if it's a child of an [<article>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/article) or [<section>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/section) it defines a specific header for that section (try not to confuse this with [titles and headings](https://developer.mozilla.org/en-US/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML#Adding_a_title)).
* [<nav>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nav) contains the main navigation functionality for the page. Secondary links, etc., would not go in the navigation.
* [<footer>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/footer) represents a group of end content for a page.

Non-semantic wrappers

Sometimes you'll come across a situation where you can't find an ideal semantic element to group some items together or wrap some content. Sometimes you might want to just group a set of elements together to affect them all as a single entity with some [CSS](https://developer.mozilla.org/en-US/docs/Glossary/CSS) or [JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript). For cases like these, HTML provides the [<div>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div) and [<span>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span) elements. You should use these preferably with a suitable [class](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes#attr-class) attribute, to provide some kind of label for them so they can be easily targeted.

[<span>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span) is an inline non-semantic element, which you should only use if you can't think of a better semantic text element to wrap your content, or don't want to add any specific meaning. For example:

<p>The King walked drunkenly back to his room at 01:00, the beer doing nothing to aid

him as he staggered through the door <span class="editor-note">[Editor's note: At this point in the

play, the lights should be down low]</span>.</p>

[<div>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div) is a block level non-semantic element, which you should only use if you can't think of a better semantic block element to use, or don't want to add any specific meaning. For example, imagine a shopping cart widget that you could choose to pull up at any point during your time on an e-commerce site:

<div class="shopping-cart">

<h2>Shopping cart</h2>

<ul>

<li>

<p><a href=""><strong>Silver earrings</strong></a>: $99.95.</p>

<img src="../products/3333-0985/thumb.png" alt="Silver earrings">

</li>

<li>

...

</li>

</ul>

<p>Total cost: $237.89</p>

</div>

Debugging