Intro to HTML and CSS - Reading 1

# HTML Introduction

HTML is the main markup language for describing the structure of web pages.



HTML stands for HyperText Markup Language. HTML is the basic building block of the World Wide Web.

Hypertext is text displayed on a computer or other electronic device with references to other text that the user can immediately access, usually by a mouse click or key press.

Apart from text, hypertext may contain tables, lists, forms, images, and other presentational elements. It is an easy-to-use and flexible format to share information over the Internet.

Markup languages use sets of markup tags to characterise text elements within a document, which gives instructions to the web browsers on how the document should appear.

HTML was originally developed by Tim Berners-Lee in 1990. He is also known as the father of the web. In 1996, the World Wide Web Consortium (W3C) became the authority to maintain the HTML specifications. HTML also became an international standard (ISO) in 2000. HTML5 is the latest version of HTML. HTML5 provides a faster and more robust approach to web development.

**Tip:** Our HTML tutorial will help you to learn the fundamentals of the latest HTML5 language, from the basic to advanced topics step-by-step. If you're a beginner, start with the basic section and gradually move forward by learning a little bit every day.

**What You Can Do with HTML**

There are a lot more things you can do with HTML.

* You can publish documents online with text, images, lists, tables, etc.
* You can access web resources such as images, videos or other HTML documents via hyperlinks.
* You can create forms to collect user inputs like name, e-mail address, comments, etc.
* You can include images, videos, sound clips, flash movies, applications and other HTML documents directly inside an HTML document.
* You can create an offline version of your website that works without the internet.
* You can store data in the user's web browser and access it later on.
* You can find the current location of your website's visitors.

The list does not end here, there are many other interesting things that you can do with HTML. You will learn about all of them in detail in upcoming chapters.

**Note:** HTML as described earlier is a markup language not a programming language, like Java, Ruby, PHP, etc. You need a web browser to view the HTML pages. The web browsers do not display the HTML tags, but uses the tags to interpret the content of the web pages.

# 

# HTML Get Started

An HTML file is simply a text file saved with an .html or .htm extension.

**Getting Started**

In this tutorial you will learn how easy it is to create an HTML document or a web page. To begin coding HTML you need only two stuff: a simple-text editor and a web browser.

Well, let's get started with creating your first HTML page.

**Creating Your First HTML Document**

Let's walk through the following steps. At the end of this tutorial, you will have made an HTML file that displays "Hello world" message in your web browser.

**Step 1: Creating the HTML file**

Open up your computer's plain text editor and create a new file.

**Tip:** Use Visual Studio Code as you’ve learned about earlier.

**Step 2: Type some HTML code**

Start with an empty window and type the following code:

**Example**

<!DOCTYPE html>

<html lang="en">

<head>

<title>A simple HTML document</title>

</head>

<body>

<p>Hello World!<p>

</body>

</html>

**Step 3: Saving the file**

Now save the file on your desktop as "myfirstpage.html ".

**Note:** It is important that the extension .html is specified — some text editors, such as Notepad, will automatically save it as .txt otherwise.

To open the file in a browser. Navigate to your file then double click on it. It will open in your default Web browser. If it does not, open your browser and drag the file to it.

**Explanation of code**

You might think what that code was all about. Well, let's find out.

* The first line <!DOCTYPE html> is the [document type declaration](https://www.tutorialrepublic.com/html-tutorial/html-doctypes.php). It instructs the web browser that this document is an HTML5 document. It is case-insensitive. Case-insensitive means that <img> is the same as <IMG>, upper or lower case letters will have the same meaning. Case-sensitive means of course the opposite in which case you need to be more careful when writing your code. But to make your code better and cleaner, please always use lower case letters for your code.
* The <head> element is a container for the tags that provides information about the document, for example, <title> tag defines the title of the document.
* The <body> element contains the document's actual content (paragraphs, links, images, tables, and so on) that is rendered in the web browser and displayed to the user.

You will learn about the different HTML elements in detail in the upcoming chapters. For now, just focus on the basic structure of the HTML document.

**Note:** A DOCTYPE declaration appears at the top of a web page before all other elements; however the doctype declaration itself is not an HTML tag. Every HTML document requires a document type declaration to insure that your pages are displayed correctly.

**Tip:** The <html>, <head>, and <body> tags make up the basic skeleton of every web page. Content inside the <head> and </head> are invisible to users with one exception: the text between <title> and </title> tags which appears as the title on a browser tab.

**HTML Tags and Elements**

HTML is written in the form of HTML elements consisting of markup tags. These markup tags are the fundamental characteristic of HTML. Every markup tag is composed of a keyword, surrounded by angle brackets, such as <html>, <head>, <body>, <title>, <p>, and so on.

HTML tags normally come in pairs like <html> and </html>. The first tag in a pair is often called the opening tag (or start tag), and the second tag is called the closing tag (or end tag).

An opening tag and a closing tag are identical, except a slash (/) after the opening angle bracket of the closing tag, to tell the browser that the command has been completed.

In between the start and end tags you can place appropriate contents. For example, a paragraph, which is represented by the p element, would be written as:

**Example**

<p>This is a paragraph.</p>

<!-- Paragraph with nested element -->

<p>

This is <b>another</b> paragraph.

</p>

# 

# HTML Preparation

URLs can be either relative or external.

**Use the examples**

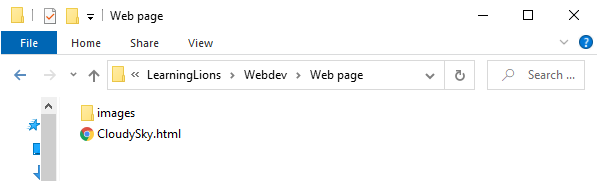
Throughout this module you will see a lot of **Examples**, it’s important that you try these out for yourself to understand how they work. However, this might not be easy as the examples might refer to files that you do not have.

**Example**

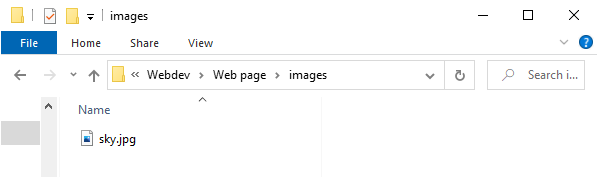
<img src="images/sky.jpg" alt="Cloudy Sky">

Here is an example when adding an image to you web page. The image, sky.jpg, should be located in the folder images, which is inside the folder of you html-file. From the name of the image, you can understand what it should depict, so just go to Google and find a substitute that you like.

So, for this example your file structure must look like this for the example to work:



Then inside the images folder you put sky.jpg.



**URLs**

When studying this module to learn HTML and CSS you’ll find that some of the code is referring to files, like images, stylesheets, web pages, PFDs, ect. These files can be referenced in different ways:

**Example**

<img src="images/sky.jpg" alt="Cloudy Sky">

The above example is a relative URL, it points to a file, which is in the same place as the web page itself, just another folder.

**Example**

<img src="https://www.coolimages.com/sky.jpg" alt="Cloudy Sky">

The above example is an absolute URL, it points to a file, which can be anywhere on the internet. With an absolute URL you can use images and files hosted on other webpages. But, make sure that images you use are free and open to use. Some have copyrights and you should be a little careful how you use them.

# 

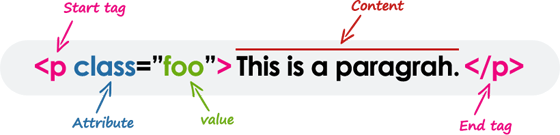
# HTML Elements

In this tutorial you will learn about HTML tags and elements.

**HTML Element Syntax**

An HTML element is an individual component of an HTML document. It represents semantics, or meaning. For example, the title element represents the title of the document.

Most HTML elements are written with a *start tag* (or opening tag) and an *end tag* (or closing tag), with content in between. Elements can also contain attributes that define its additional properties. For example, a paragraph, which is represented by the p element, would be written as:



We will learn about the HTML attributes in the [next chapter](https://www.tutorialrepublic.com/html-tutorial/html-attributes.php).

**Note:** All elements don't require the end tag or closing tag to be present. These are referred as *empty elements*, *self-closing elements* or *void elements*.

**HTML Tags Vs Elements**

Technically, an HTML element is the collection of start tags, its attributes, an end tag and everything in between. On the other hand an HTML tag (either opening or closing) is used to mark the start or end of an element, as you can see in the above illustration.

However, in common usage the terms HTML element and HTML tag are interchangeable i.e. a tag is an element is a tag. For simplicity's sake on this website, the terms "tag" and "element" are used to mean the same thing — as it will define something on your web page.

**Case Insensitivity in HTML Tags and Attributes**

In HTML, tag and attribute names are NOT case-sensitive (but most attribute values are case-sensitive). It means the tag <P>, and the tag <p> defines the same thing in HTML which is a paragraph.

**Example**

<p>This is a paragraph.</p>

<P>This is also a valid paragraph.</P>

Both these elements will work. But for a clean code we will always try to use lower case letters.

**Tip:** We recommend using lowercase for tag and attributing names in HTML, since by doing this you can make your document more compliant for future upgrades.

**Empty HTML Elements**

Empty elements (also called self-closing or void elements) are not container tags — that means, you can not write <hr>*some content*</hr> or <br>*some content*</br>.

A typical example of an empty element, is the <br> element, which represents a line break. Some other common empty elements are <img>, <input>, <link>, <meta>, <hr>, etc.

**Example**

<p>This paragraph contains <br> a line break.</p>

<img src="images/sky.jpg" alt="Cloudy Sky">

<input type="text" name="username">

**Note:** In HTML, a self-closing element is written simply as [<br>](https://www.tutorialrepublic.com/html-reference/html-br-tag.php). In XHTML, a self-closing element requires a space and a trailing slash, such as <br />.

**Nesting HTML Elements**

Most HTML elements can contain any number of further elements (except [empty elements](https://www.tutorialrepublic.com/html-tutorial/html-elements.php#empty-elements)), which are, in turn, made up of tags, attributes, and content or other elements.

The following example shows some elements nested inside the <p> element.

**Example**

<p>Here is some <b>bold</b> text.</p>

<p>Here is some <em>emphasized</em> text.</p>

<p>Here is some <mark>highlighted</mark> text.</p>  
  
<p>Click <a href="https://Google.com">here</a> to open Google</p>  
  
<div>  
 <p>This text is inside a div-element</p>  
</div>

**Tip:** Placing one element inside another is called nesting. A nested element, also called a child element, can be a parent element too if other elements are nested within it.

HTML tags should be nested in correct order. They must be closed in the inverse order of how they are defined, that means the last tag opened must be closed first.

**Example**

<p><strong>These tags are nested properly.</strong></p>

<p><strong>These tags are not nested properly.</p></strong>

# 

# HTML Elements Types

Elements can be placed in two distinct groups: [*block level*](https://www.tutorialrepublic.com/css-tutorial/css-visual-formatting.php#block-level) and [*inline level*](https://www.tutorialrepublic.com/css-tutorial/css-visual-formatting.php#inline-level) elements. The former make up the document's structure, while the latter dress up the contents of a block.

Also, a block element occupies 100% of the available width and it is rendered with a line break before and after. Whereas, an inline element will take up only as much space as it needs.

The most commonly used block-level elements are <div>, <p>, <h1> through <h6>, <form>, <ol>, <ul>, <li>, and so on. Whereas, the commonly used inline-level elements are <img>, <a>, <span>, <strong>, <b>, <em>, <i>, <code>, <input>, <button>, etc.

You will learn about these elements in detail in upcoming chapters.

**Note:** The block-level elements should not be placed within inline-level elements. For example, the <p> element should not be placed inside the <b> element.

**HTML Attributes**

In this tutorial you will learn how to use attributes to give more meaning to HTML tags.

**What are Attributes**

Attributes define additional characteristics or properties of the element such as width and height of an image. Attributes are always specified in the start tag (or opening tag) and usually consists of name/value pairs like name="value". Attribute values should always be enclosed in quotation marks.

Also, some attributes are required for certain elements. For instance, an <img> tag must contain a src and alt attributes. Let's take a look at some examples of the attributes usages:

**Example**

<img src="images/smiley.png" width="30" height="30" alt="Smiley">

<a href="https://www.google.com/" title="Search Engine">Google</a>

<abbr title="Hyper Text Markup Language">HTML</abbr>

<input type="text" value="John Doe">

In the above example src inside the <img> tag is an attribute and image path provided is its value. Similarly href inside the <a> tag is an attribute and the link provided is its value, and so on.

**Tip:** Both single and double quotes can be used to quote attribute values. However, double quotes are most common. In situations where the attribute value itself contains double quotes it is necessary to wrap the value in single quotes, e.g., value='John "Williams" Jr.'

There are several attributes in HTML5 that do not consist of name/value pairs but consist of just names. Such attributes are called Boolean attributes. As you remember Boolean values is something that is either **True** or **False**. When we talk about Boolean attributes in HTML, it's very similar:  
**TRUE**: The HTML element has the attribute and its effect is On/Activated  
**FALSE**: The HTML element don’t have the attribute and its effect is Off/Disable  
  
Examples of some commonly used Boolean attributes are checked, disabled, readonly, required, etc.

**Example**

<input type="email" required>

<input type="submit" value="Submit" disabled>

<input type="checkbox" checked>

<input type="text" value="Read only text" readonly>

You will learn about all these elements in detail in upcoming chapters.

**Note:** Attribute values are generally case-insensitive, except certain attribute values, like the id and class attributes. However, the World Wide Web Consortium (W3C) recommends lowercase for attribute values in their specification.

**General Purpose Attributes**

There are some attributes, such as id, title, class, style, etc. that you can use on the majority of HTML elements. The following section describes their usages.

Some of the attributes and their purpose are linked to styling and CSS, which you will learn more about later.

**The id Attribute**

The id attribute is used to give a unique name or identifier to an element within a document. This makes it easier to select the element using CSS or JavaScript.

**Example**

<input type="text" id="firstName">

<div id="container">Some content</div>

<p id="infoText">This is a paragraph.</p>

**Note:** The id of an element must be unique within a single document. No two elements in the same document can be named with the same id, and each element can have only one id.

**The class Attribute**

Like id attribute, the class attribute is also used to identify elements. But unlike id, the class attribute does not have to be unique in the document. This means you can apply the same class to multiple elements in a document, as shown in the following example:

**Example**

<input type="text" class="highlight">

<div class="box highlight">Some content</div>

<p class="highlight">This is a paragraph.</p>

**Tip:** Since a class can be applied to multiple elements, therefore any style rules that are written to that class will be applied to all the elements having that class.

**The title Attribute**

The title attribute to is used to provide advisory text about an element or its content. Try out the following example to understand how this actually works.

**Example**

<abbr title="World Wide Web Consortium">W3C</abbr>

<a href="images/kites.jpg" title="Click to view a larger image">

<img src="images/kites-thumb.jpg" alt="kites">

</a>

**Tip:** The value of the title attribute (i.e. title text) is displayed as a tooltip by the web browsers when the user place mouse cursor over the element.

**The style Attribute**

The style attribute allows you to specify CSS styling rules such as color, font, border, etc. directly within the element. Let's check out an example to see how it works:

**Example**

<p style="color: blue;">This is a paragraph.</p>

<img src="images/sky.jpg" style="width: 300px;" alt="Cloudy Sky">

<div style="border: 1px solid red;">Some content</div>

The attributes we've discussed above are also called global attributes.

# 

# HTML Headings

In this tutorial you will learn how to create headings in HTML.

**Organizing Content with Headings**

Headings help in defining the hierarchy and the structure of the web page content.

HTML offers six levels of heading tags, <h1> through <h6>; the higher the heading level number, the greater its importance — therefore <h1> tag defines the most important heading, whereas the <h6> tag defines the least important heading in the document.

By default, browsers display headings in larger and bolder font than normal text. Also, <h1> headings are displayed in largest font, whereas <h6> headings are displayed in smallest font.

**Example**

<h1>Heading level 1</h1>

<h2>Heading level 2</h2>

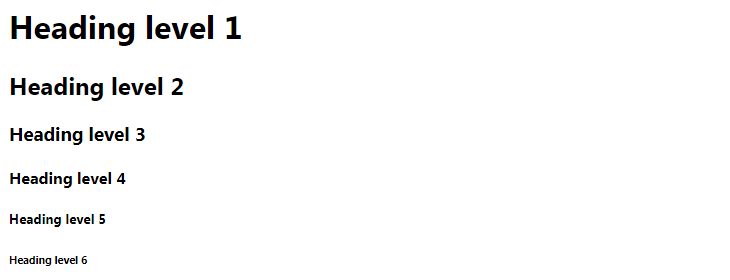
<h3>Heading level 3</h3>

<h4>Heading level 4</h4>

<h5>Heading level 5</h5>

<h6>Heading level 6</h6>

— The output of the above example will look something like this:



**Note:** Each time you place a heading tag on a web page, the web browser built-in style sheets automatically create some empty space (called margin) before and after each heading. You can use the CSS [margin](https://www.tutorialrepublic.com/css-reference/css-margin-property.php) property to override the browser's default style sheet.

**Tip:** You can easily customise the appearance of HTML heading tags such as their font size, boldness, typeface, etc. using the CSS [font](https://www.tutorialrepublic.com/css-reference/css-font-property.php) properties.

**Importance of Headings**

* HTML headings provide valuable information by highlighting important topics and the structure of the document, so optimise them carefully to improve user engagement.
* Don't use headings to make your text look BIG or bold. Use them only for highlighting the heading of your document and to show the document structure.
* Since search engines, such as Google, use headings to index the structure and content of the web pages, use them very wisely in your webpage.
* Use the <h1> headings as main headings of your web page, followed by the <h2> headings, then the less important <h3> headings, and so on.

**Tip:** Use the <h1> tag to mark the most important heading which is usually at the top of the page. An HTML document generally should have exactly one <h1> heading, followed by the lower-level headings such as <h2>, <h3>, <h4>, and so on.

# 

# HTML Paragraphs

In this tutorial you will learn how to create paragraphs in HTML.

**Creating Paragraphs**

Paragraph element is used to publish text on the web pages.

Paragraphs are defined with the <p> tag. Paragraph tag is a very basic and typically the first tag you will need to publish your text on the web pages. Here's an example:

**Example**

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

**Note:** Browsers built-in style sheets automatically create some space above and below the content of a paragraph (called [margin](https://www.tutorialrepublic.com/css-reference/css-margin-property.php)), but you can override it using CSS.

**Creating Line Breaks**

The <br> tag is used to insert a line break on the web page.

Since the <br> is an [empty element](https://www.tutorialrepublic.com/html-tutorial/html-elements.php#empty-elements), so there is no need of corresponding </br> tag.

**Example**

<p>This is a paragraph <br> with line breaks.</p>

<p>This is <br>another paragraph <br> with line breaks.</p>

**Note:** Another correct usage for line breaks is to use the self-closing element <br />

**Note:** Don't use the empty paragraph i.e. <p></p> to add extra space in your web pages. The browser may ignore the empty paragraphs since it is a logical tag. Use the CSS [margin](https://www.tutorialrepublic.com/css-reference/css-margin-property.php) property instead to adjust the space around the elements.

**Creating Horizontal Rules**

You can use the <hr> tag to create horizontal rules or lines to visually separate content sections on a web page. Like <br>, the <hr> tag is also an empty element. Here's an example:

**Example**

<p>This is a paragraph.</p>

<hr>

<p>This is another paragraph.</p>

**Managing White Spaces**

Normally the browser will display the multiple spaces created inside the HTML code by pressing the *space-bar key* or *tab key* on the keyboard as a single space. Multiple line breaks created inside the HTML code through pressing the enter key are also displayed as a single space.

The following paragraphs will be displayed in a single line without any extra space:

**Example**

<p>This paragraph contains multiple spaces in the source code.</p>

<p>

This paragraph

contains multiple tabs and line breaks

in the source code.

</p>

Insert &nbsp; for creating extra consecutive spaces, while insert <br> tag for creating line breaks on your web pages, as demonstrated in the following example:

**Example**

<p>This paragraph has multiple &nbsp;&nbsp;&nbsp;spaces.</p>

<p>This paragraph has multiple<br><br>line<br><br><br>breaks.</p>

**Defining Preformatted Text**

Sometimes, using &nbsp;, <br>, etc. for managing spaces isn't very convenient. Alternatively, you can use the <pre> tag to display spaces, tabs, line breaks, etc. exactly as written in the HTML file. It is very helpful in presenting text where spaces and line breaks are important like poems or code.

The following example will display the text in the browser as it is in the source code:

**Example**

<pre>

Twinkle, twinkle, little star,

How I wonder what you are!

Up above the world so high,

Like a diamond in the sky.

</pre>

**Tip:** Text within the <pre> element is typically rendered by the browsers in a monospace or fixed-width font, such as Courier, but you can override this using the CSS [font](https://www.tutorialrepublic.com/css-reference/css-font-property.php) property.

# 

# HTML Text Formatting

In this tutorial you will learn how to format the text on the web pages using HTML tags.

**Formatting Text with HTML**

HTML provides several tags that you can use to make some text on your web pages to appear differently than normal text, for example, you can use the tag <b> to make the text bold, tag <i> to make the text italic, tag <mark> to highlight the text, tag <code> to display a fragment of computer code, tags <ins> and <del> for marking editorial insertions and deletions, and more.

The following example demonstrates the most commonly used formatting tags in action. Now, let's try this out to understand how these tags basically work:

**Example**

<p>This is <b>bold text</b>.</p>

<p>This is <strong>strongly important text</strong>.</p>

<p>This is <i>italic text</i>.</p>

<p>This is <em>emphasized text</em>.</p>

<p>This is <mark>highlighted text</mark>.</p>

<p>This is <code>computer code</code>.</p>

<p>This is <small>smaller text</small>.</p>

<p>This is <sub>subscript</sub> and <sup>superscript</sup> text.</p>

<p>This is <del>deleted text</del>.</p>

<p>This is <ins>inserted text</ins>.</p>

By default, the <strong> tag is typically rendered in the browser as <b>, whereas the <em> tag is rendered as <i>. However, there is a difference in the meaning of these tags.

**Difference between <strong> and <b> tag**

Both <strong> and <b> tags render the enclosed text in a bold typeface by default, but the <strong> tag indicates that its contents have strong importance, whereas the <b> tag is simply used to draw the reader's attention without conveying any special importance.

**Example**

<p><strong>WARNING!</strong> Please proceed with caution.</p>

<p>The concert will be held at <b>Hyde Park</b> in London.</p>

**Difference between <em> and <i> tag**

Similarly, both <em> and <i> tags render the enclosed text in italic type by default, but the <em> tag indicates that its contents have stressed emphasis compared to surrounding text, whereas the <i> tag is used for marking up text that is set off from the normal text for readability reasons, such as a technical term, an idiomatic phrase from another language, a thought, etc.

**Example**

<p>Cats are <em>cute</em> animals.</p>

<p>The <i>Royal Cruise</i> sailed last night.</p>

**Note:** Use the [<em>](https://www.tutorialrepublic.com/html-reference/html-em-tag.php) and [<strong>](https://www.tutorialrepublic.com/html-reference/html-strong-tag.php) tags when the content of your page requires that certain words or phrases should have strong emphasis or importance. Also, in HTML5 the <b> and <i> tags have been redefined, earlier they don't have semantic meaning.

**Showing Abbreviations**

An abbreviation is a shortened form of a word, phrase, or name.

You can use the <abbr> tag to denote an abbreviation. The title attribute is used inside this tag to provide the full expansion of the abbreviation, which is displayed by the browsers as a tooltip when the mouse cursor is hovered over the element. Let's try out an example:

**Example**

<p>The <abbr title="World Wide Web Consortium">W3C</abbr> is the main international standards organisation for the <abbr title="World Wide Web">WWW or W3</abbr>. It was founded by Tim Berners-Lee.</p>

**Marking Contact Addresses**

Web pages often include street or postal addresses. HTML provides a special tag <address> to represent contact information (physical and/or digital) for a person, people or organisation.

This tag should ideally be used to display contact information related to the document itself, such as the article's author. Most browsers display an address block in italic. Here's an example:

**Example**

<address>

Mozilla Foundation<br>

331 E. Evelyn Avenue<br>

Mountain View, CA 94041, USA

</address>

Please check out the HTML reference section for a complete list of [HTML formatting tags](https://www.tutorialrepublic.com/html-reference/html5-tags.php#formatting-tags).

# 

# Best Practices to Write Clean HTML Code

Have you ever looked at your HTML and wondered…

Am I writing good code?!

Does it look professional enough? Am I doing things right? Is this how clean HTML code looks like?

**Having those questions is normal and good,**specially if you don’t wanna end up writing messy, jumbled, sloppy, spaghetti code that looks like:

Writing clean code should be one of your biggest concerns because most web pages are presented via HTML. You don’t want to mess up the foundations of your site. It’s just wrong and embarrassing.

Here you’ll find some great practices to forget about those doubts and write clean correct HTML that will impress everyone who reads it.  
  
1. Give a f\*\*k

Seriously. The most important thing you need to write good code is your desire to *do things right*. Writing clean HTML code is easy, but you need to care and pay a lot of attention.

“Clean code is a code that is written by someone who cares” — Michael Feathers

**Your code reflects the kind of developer you are.**You can tell if someone is passionate about coding or if he doesn’t give a f\*\*k just by reading a few lines of his HTML code. It’s not a coincidence that lazy unprofessional people are the ones who write bad and unorganized code.

2. Indent

Indented code is easier to read, easier to understand, easier to modify, and easier to maintain. Just take a look at how difficult is to understand the parent-child relationships between elements when the code isn’t indented correctly:

**Bad HTML Code:**   
<footer id="footer-bottom">  
<div class="container">  
 <div class="row">  
 <div id="footer-copyrights" class="col-md-6">  
<p> &copy; 2018 All Rights Reserved.</p>  
 </div>  
 <div id="footer-menu" class="col-md-6">  
 <ul><li>Home</li>  
 <li>Team</li>  
 <li>Pricing</li></ul>  
</div>  
</div>  
</div>  
 </footer>  
  
**Better HTML Code:**<footer id="footer-bottom">  
 <div class="container">  
 <div class="row">  
 <div id="footer-copyrights" class="col-md-6">  
 <p> &copy; 2018 All Rights Reserved.</p>  
 </div>  
 <div id="footer-menu" class="col-md-6">  
 <ul>  
 <li>Home</li>  
 <li>Team</li>  
 <li>Pricing</li>  
 </ul>  
 </div>  
 </div>  
 </div>  
</footer>

How many spaces should you use for indentation? Many people say 4 spaces, [many others](https://google.github.io/styleguide/htmlcssguide.html) say 2, and almost everyone hates tabs.

My advice? **Don’t over think it, it’s not really important to use a specific space value.** What really matters is…

**3. Just Be Consistent**

Writing clean HTML code isn’t always about choosing good practices and avoiding bad ones. Many times you can use different approaches to write the same line of code.

For example:

* You can use dash or underscore for ids and classes
* You can use single quote or double quote argument
* You can indent with 2 or 4 spaces

Just be consistent. **When you find a specific practice that you like, stick to it and use it everywhere.** Inconsistency makes your code extremely confusing for you and for the poor soul who has to read it.

4. Exterminate “Divitis”

[“Divitis”](https://csscreator.com/divitis) is a funny name for a common problem in HTML. We often overuse divs, mainly because we want to wrap and target all elements for styling in CSS. Then we end up having an eternal list of divs that are completely unnecessary:

**Bad HTML Code:**<div id="content">  
 <div class="headline">Headline</div>  
 <div class="subtitle">Subtitle</div>  
 <div class="post">Post Content</div>  
 <div class="list">   
 <ul>   
 <li>Element 1</li>   
 <li>Element 2</li>  
 </ul>   
 </div>  
</div> <!-- end content div -->  
  
**Better HTML Code:**<div id="content">  
 <h1>Headline</h1>  
 <h2>Subtitle</h2>  
 <p>Post Content</p>  
 <ul>   
 <li>Element 1</li>   
 <li>Element 2</li>  
 </ul>   
</div>

Just as writers edit their draft many times during the writing process, **you should constantly refactor or rewrite your HTML.**Reduce the number of divs and clean up your code deleting all the unnecessary elements.

5. Avoid Comments

HTML is not a programming language, **comments aren’t necessary because HTML markup is very much self-explanatory.**If you find yourself commenting your HTML a lot, you should review the [HTML elements reference](https://developer.mozilla.org/en-US/docs/Web/HTML/Element).

“Don’t comment on bad code. Rewrite it.” — Brian W. Kernighan

Your code won’t be easier to understand just because you add comments everywhere. You can use them to make things a bit more clear (for example, to show you’re closing a div after many lines of code). But don’t comment on things that are obvious or code that is badly written.

6. Class = “clear-name”

Classes will come more into play later when you start to learn about CSS and styling.

What does *class=*“*wpr”* means?

Width Paragraph? WordPress? Whopper?! Clean HTML code shouldn’t look like a trivia game.

**Use meaningful names for your Ids and Classes. They should be short, descriptive and represent only one concept.** It’ll make your HTML clearer and the styling process easier.

Don’t you think it’d be better if you named your class “*wrapper”* instead of “*wpr”*?

7. Use Whitespace

Many people write smashed-together code without using white spaces. The result? It’s like reading a book with no punctuation or paragraphs:

**Hell:**

<body>  
<h1>Table</h1><table><tr><th>Data 1</th><th>Data 2</th></tr>  
<tr><td>Calcutta</td><td>Orange</td></tr></table></body>  
  
**Better Code:**<body><h1>Table</h1><table>  
 <tr>  
 <th>Data 1</th>  
 <th>Data 2</th>  
 </tr>  
 <tr>  
 <td>Calcutta</td>  
 <td>Orange</td>  
 </tr>  
</table></body>

**Use whitespace: indentation, empty lines, line breaks.** Other humans (including you) should be able to read your code in the future without having a headache.