<https://www.freecodecamp.org/news/html-basics-for-beginners/>

Contents

[HTML 3](#_Toc116567748)

[What Is HTML? 3](#_Toc116567749)

[What Are HTML Elements? 3](#_Toc116567750)

[How to Nest HTML Elements 4](#_Toc116567751)

[What are HTML Attributes? 5](#_Toc116567752)

[Common HTML elements 5](#_Toc116567753)

[Block-level vs inline HTML elements 6](#_Toc116567754)

[How to comment in HTML 7](#_Toc116567755)

[How to use HTML entities 7](#_Toc116567756)

[How to use emoji in HTML 8](#_Toc116567757)

[Common beginner mistakes in HTML 8](#_Toc116567758)

[1. Tags/Element names 8](#_Toc116567759)

[2. Closing tag 8](#_Toc116567760)

[3. Nesting 8](#_Toc116567761)

[4. Single quotes and Double quotes 8](#_Toc116567762)

[How to build a simple website with HTML 8](#_Toc116567763)

[How to create an HTML document 8](#_Toc116567764)

[How to build a pancake recipe page 9](#_Toc116567765)

[Conclusion 14](#_Toc116567766)

[HTML5 Semantic Elements 15](#_Toc116567767)

[What are Semantic Elements? 15](#_Toc116567768)

[List of new semantic elements 15](#_Toc116567769)

[Why use semantic elements? 16](#_Toc116567770)

[<section> and <article> 17](#_Toc116567771)

[<header> and <hgroup> 17](#_Toc116567772)

[<aside> 18](#_Toc116567773)

[<footer> 19](#_Toc116567774)

[<small> 19](#_Toc116567775)

[<time> 19](#_Toc116567776)

[<figure> and <figcaption> 19](#_Toc116567777)

[HTML Metadata 20](#_Toc116567778)

[How to set up an HTML project 20](#_Toc116567779)

[What are meta tags in HTML? 21](#_Toc116567780)

[How to define the character set of a website 21](#_Toc116567781)

[How to let Microsoft's Internet Explorer know which rendering view to use 22](#_Toc116567782)

[How to adjust viewport settings 22](#_Toc116567783)

[Additional **meta** tags to add to your HTML project 23](#_Toc116567784)

[How to add a description of your webpage 23](#_Toc116567785)

[How to add the name of the website's author 24](#_Toc116567786)

[Conclusion 24](#_Toc116567787)

[Web Accessibility 25](#_Toc116567788)

[Semantic HTML 25](#_Toc116567789)

[Image Alt Text 27](#_Toc116567790)

[Well-Structured HTML 28](#_Toc116567791)

[The HTML lang Attribute 28](#_Toc116567792)

[Descriptive Hyperlinks 29](#_Toc116567793)

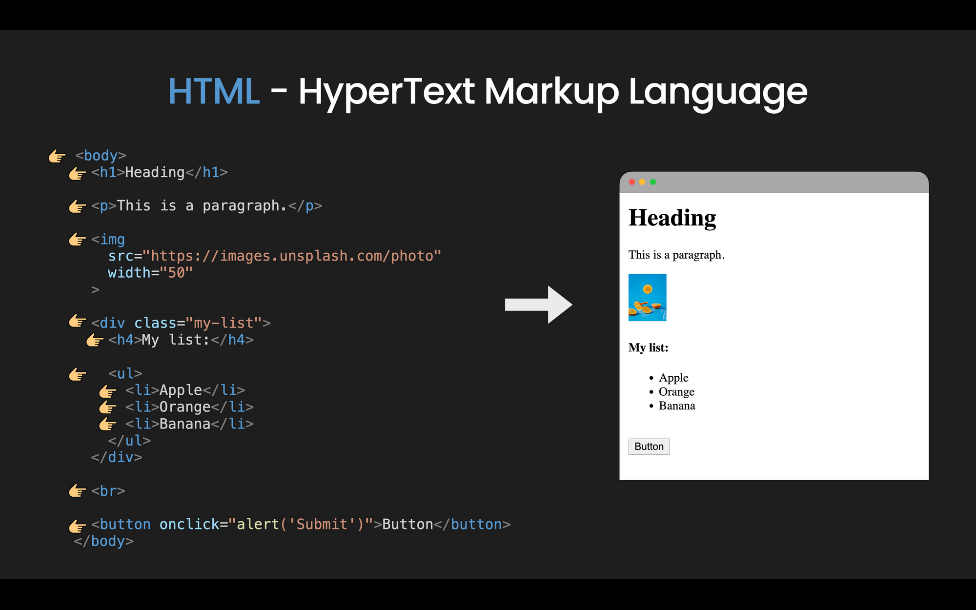
[ARIA Roles 29](#_Toc116567794)

[Accessibility Starts With Your HTML 31](#_Toc116567795)

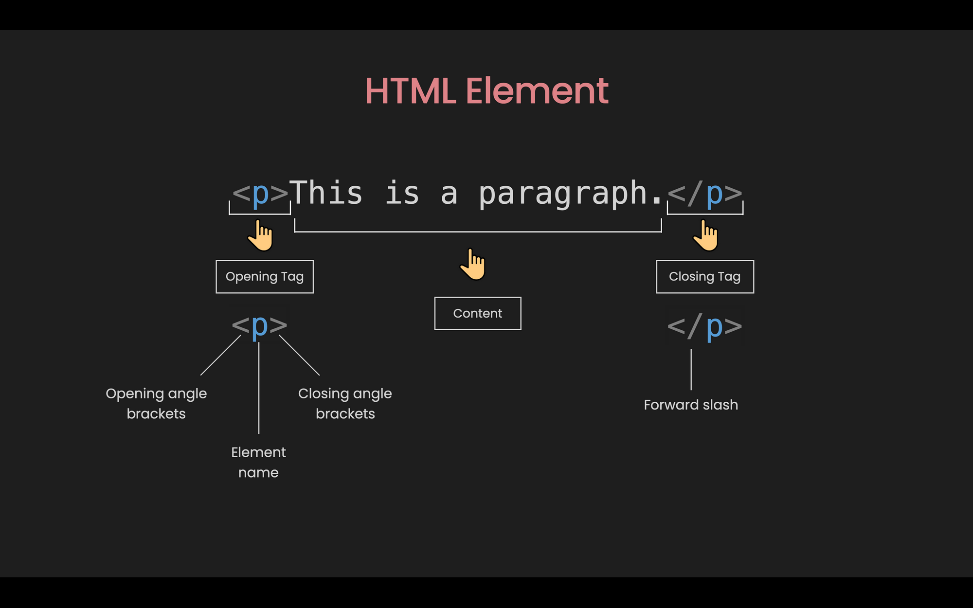
# HTML

## What Is HTML?

HTML, which stands for Hypertext Markup Language, is a pretty simple language. It consists of different elements which we use to structure a web page.



## What Are HTML Elements?



The element usually starts with an opening tag, which consists of the name of the element. It's wrapped in opening and closing angle brackets. The opening tag indicates where the element begins.

Similar to the opening tag, the closing tag is also wrapped in opening and closing angle brackets. But it also includes a forward slash before the element's name.

Everything inside the opening and closing tags is the content.

But not all elements follow this pattern. We call those that don't empty elements. They only consist of a single tag or an opening tag that cannot have any content. These elements are typically used to insert or embed something in the document.

For example, the <img> element is used to embed an image file, or the <input> element is used to insert an input onto the page.

<img src="https://images.unsplash.com/photo-1610447847416-40bac442fbe6" width="50">

In the example above, the <img> element only consists of one tag that does not have any content. This element is used to insert an image file from [Unsplash](https://unsplash.com/) in the document.

## How to Nest HTML Elements

<div class="my-list">

<h4>My list:</h4>

<ul>

<li>Apple</li>

<li>Orange</li>

<li>Banana</li>

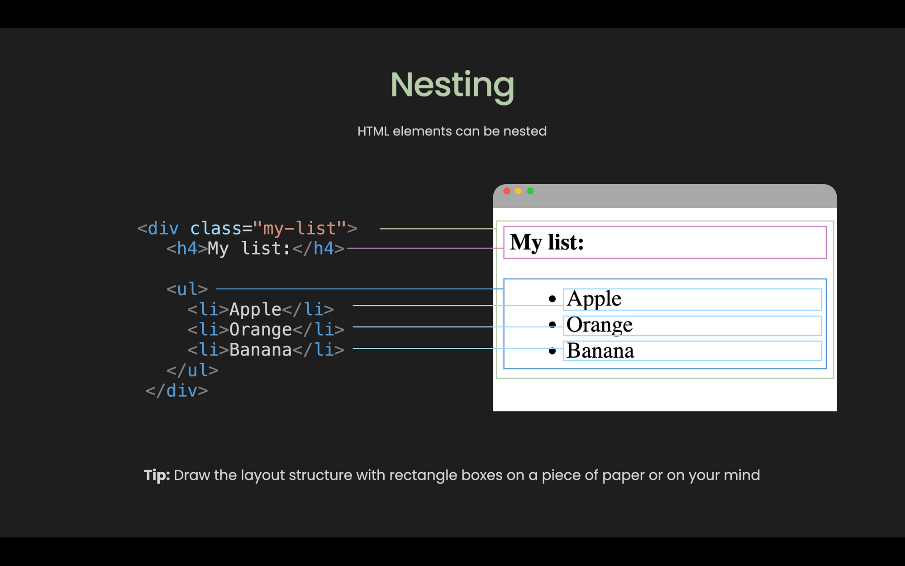
</ul>

</div>

Elements can be placed inside other elements. This is called Nesting. In the example above, inside the <div> element we have an <h4> element and an <ul> or unordered list element. And Similarly inside the <ul> element, there are 3 <li> or list item elements.

Basic nesting is quite straight-forward to understand. But when the page gets larger, nesting can become complicated.

Therefore, before working with HTML, think about the layout structure you would like to have. You can draw it out on a piece of paper or in your mind. It will help a lot.

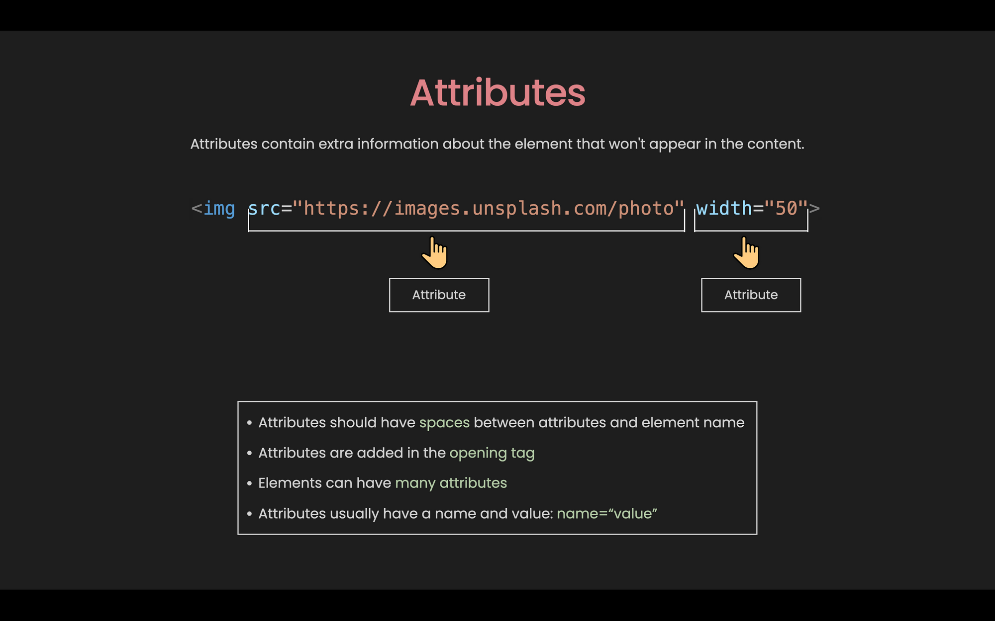


## What are HTML Attributes?

Elements also have attributes, which contain extra information about the element that will not appear in the content.

<img src="https://images.unsplash.com/photo" width="50">

In the example above, the <img> element has 2 attributes: src or source to specify the path of the image, and width to specify the width of the image in pixels.



With this example, you can see the following characteristics of attributes:

* There is a space between attributes and the element name
* Attributes are added in the opening tag
* Elements can have many attributes
* Attributes usually have a name and a value: name=“value”

But not every attribute has the same pattern. Some can exist without values, and we call them Boolean Attributes.

<button onclick=“alert('Submit')" disabled>Button</button>

In this example, if we want to disable the button, all we have to do is pass a disabled attribute without any values. This means that the presence of the attribute represents the true value, otherwise, the absence represents the false value.

### Common HTML elements

There are in total more than 100 elements. But 90% of the time you will only use around 20 of the most common. I have put them into 5 groups:

#### Section elements

<div>, <span>, <header>, <footer>, <nav>, <main>, <section>

These elements are used to organize the content into different sections. They are usually self-explanatory, for example, <header> usually represents a group of the introduction and navigation section, <nav> represents the section that contains navigation links, and so on.

#### Text content

<h1> to <h6>, <p>, <div>, <span>, <ul>, <ol>, <li>

These elements are used to organize content or text blocks. They are important to accessibility and SEO. They tell the browser the purpose or structure of the content.

#### Forms

<form>, <input>, <button>, <label>, <textarea>

These elements can be used together to create forms that users can fill out and submit. Forms might be the trickiest part of HTML.

#### Images and Links

<img>, <a>

These elements are used to insert an image or create a hyperlink.

#### Others

<br>, <hr>

These elements are used to add a break to the webpage.

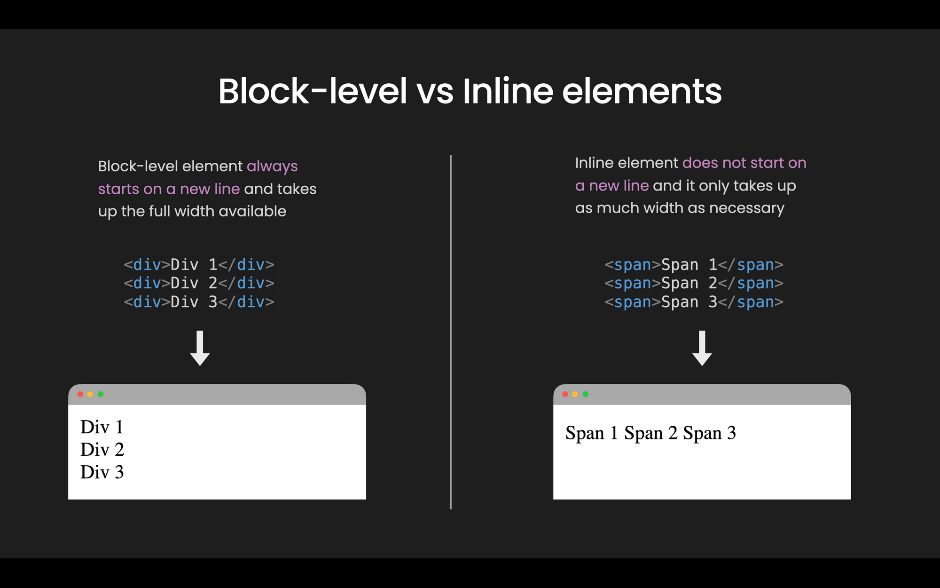
You can find all the elements on [developer.mozilla.org](https://developer.mozilla.org/en-US/docs/Web/HTML/Element). But for beginners, you just need to know the most common ones.

## Block-level vs inline HTML elements

By default, an element can be either block-level or an inline element.

Block-level elements are the elements that always start on a new line and take up the full width available.

Inline elements are the elements that do not start on a new line and it only take up as much width as necessary.



Two elements that represent block-level and inline elements, respectively, are <div> and <span>. In this example, you can see that the <div> elements takes 3 lines, whereas the <span> element only takes up 1 line.

But the question is: how do we know which ones are block-level elements and which ones are inline elements? Well, unfortunately you need to remember them. The easiest way is to remember which are inline elements – and the rest are block elements.

If we look back at the most common HTML elements, inline elements include: <span>, <input>, <button>, <label>, <textarea>, <img>, <a>, <br>.

## How to comment in HTML

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

<!-- <p>I am not showing.</p> -->

The purpose of comments is to include notes in the code to explain your logic or simply to organize your code.

HTML comments are wrapped in the special markers: <!-- and --> and they are ignored in the browser.

## How to use HTML entities

What if you want to show the text: the <p> tag defines a paragraph., but the browser interprets <p> as an opening tag for a new element? In this case, we can use HTML entities like in the following example:

<p>the <p> tag defines a paragraph.</p>

<p>the &lt;p&gt; define a paragraph.</p>

## How to use emoji in HTML

In the modern web, we can display emoji in HTML pretty easily, like this: 👻

<p>😀 Grinning Face.</p>

<p>🎂 Birthday</p>

## Common beginner mistakes in HTML

### 1. Tags/Element names

Tags/Element names are cAse-inSensitive. This means that they can be written in lowercase or uppercase, but it is recommended that you write everything in lowercase: <button> not <ButTon>.

### 2. Closing tag

Failing to include a closing tag is a common beginner error. Therefore, whenever you create an opening tag, immediately put in a closing tag.

### 3. Nesting

This is wrong:

<div>Div 1 <span> Span 2 </div></span>

The tags have to open and close in a way that they are inside or outside one another.

### 4. Single quotes and Double quotes

This is wrong:

<img src="https://images.unsplash.com/'>

You cannot mix single quotes and double-quotes. You should always use double quotes and use HTML entities if needed.

## How to build a simple website with HTML

Individual HTML elements are not enough to create a website. So let's see what more we need to build a simple website from scratch.

### How to create an HTML document

First, let's open [Visual Studio Code](https://code.visualstudio.com/) (or your favorite code editor). In the folder of your choice, create a new file and name it index.html.

In the index.html file, type ! (exclamation mark) and press enter. You will see something like this:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

</body>

</html>

This is the minimal code that an HTML document should have to make up a website. And here we have:

1. <!DOCTYPE html>: First we have Doctype. For some weird historical reason in HTML we have to include the doctype for everything to work correctly.
2. <html lang="en"></html>: The <html> element wraps all the content on the page, also known as the root element. And we should always include the lang attribute to declare the language of the page.
3. <head></head>: The <head> element is a container for everything you want to include, but not content that you show to your users.
4. <meta charset="UTF-8" />: The first meta element is used to set the character set to be UTF-8, which includes most characters from written languages.
5. <meta name="viewport" content="width=device-width, initial-scale=1.0" />: The second meta element specifies the browser viewport. This setting is for a mobile-optimized site.
6. <title>Document</title>: This is the <title> element. It sets the title of the page.
7. <body></body>: The <body> element contains all the content on the page.

### How to build a pancake recipe page

Alright, now that we have the starter code, let's build a pancake recipe page. We are going to use the content from this [AllRecipes Page](https://www.allrecipes.com/recipe/21014/good-old-fashioned-pancakes/).

First, let's give the <title> element content of the pancakes recipe. You will see the text on the web page tab change. In the <body> element, let's create 3 elements: <header>, <main> and <footer> representing 3 sections.

#### 1. Build the header section

In the header, we want to have the logo and the navigation. Therefore, let's create a div with the content ALL RECIPE for the logo.

For the navigation, let's use the <nav> element. Within the <nav> element, we can use <ul> to create an unordered list. We want to have 3 <li> elements for 3 links: Ingredients, Steps, and Subscribe. The header code looks like this:

...

<header>

<div>ALL RECIPE</div>

<nav>

<ul>

<li><a href="#ingredients">Ingredients</a></li>

<li><a href="#steps">Steps</a></li>

<li><a href="#subsribe">Subscribe</a></li>

</ul>

</nav>

</header>

...

#### 2. Build the Main Section

In the main section, first, we want to have a title and an image. We can use h1 for the title and <img> for the image (we can use an image from [Unsplash](https://images.unsplash.com/) for free):

...

<main>

<h1>Good Old Fashioned Pancakes</h1>

<img

src="https://images.unsplash.com/photo-1575853121743-60c24f0a7502"

alt="pancake"

width="250"

/>

</main>

...

Next, we want to list all the ingredients. We can use <ol> to create an ordered list and <input type="checkbox" /> to create a checkbox.

But before that, we can use <h2> to start a new content block. We also want to add the id attribute for <h2> so that the link in the navigation knows where to go:

...

<main>

...

<h2 id="ingredients">Ingredients</h2>

<ol>

<li><input type="checkbox" /> 1 ½ cups all-purpose flour</li>

<li><input type="checkbox" /> 3 ½ teaspoons baking powder</li>

<li><input type="checkbox" /> 1 teaspoon salt</li>

<li><input type="checkbox" /> 1 tablespoon white sugar</li>

<li><input type="checkbox" /> 1 ¼ cups milk</li>

<li><input type="checkbox" /> 1 egg</li>

</ol>

</main>

...

After the ingredients, we want to list all the steps. We can use <h4> for the step heading and <p> for the step content:

...

<main>

...

<h2 id="steps">Steps</h2>

<h4>Step 1</h4>

<p>

In a large bowl, sift together the flour, baking powder, salt and sugar.

Make a well in the center and pour in the milk, egg and melted butter;

mix until smooth.

</p>

<h4>Step 2</h4>

<p>

Heat a lightly oiled griddle or frying pan over medium-high heat. Pour

or scoop the batter onto the griddle, using approximately 1/4 cup for

each pancake. Brown on both sides and serve hot.

</p>

</main>

...

Alright, now that we are done with the main section, let's move on to the footer section.

#### 3. Build the Footer Section

In the footer, we want to have a subscribe form and copyright text.

For the subscribe form, we can use the <form> element. Inside it, we can have an <input type="text"> for text input and a <button> for the submit button.

For the copyright text, we can simply use a <div>. Notice here, we can use the HTML entity $copy; for the copyright symbol.

We can add <br> to add some space between the subscribe form and the copyright text:

...

<footer>

<h6 id="subscribe">Subscribe</h6>

<form onsubmit="alert('Subscribed')">

<input type="text" placeholder="Enter Email Address" />

<button>Submit</button>

</form>

<br />

<div>&copy; dakota kelly at Allrecipe.com</div>

</footer>

...

#### Full code

Alright now we are done! Here is the full code for reference:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Pancake Recipe</title>

</head>

<body>

<header>

<div>ALL RECIPE</div>

<nav>

<ul>

<li><a href="#ingredients">Ingredients</a></li>

<li><a href="#steps">Steps</a></li>

<li><a href="#subscribe">Subscribe</a></li>

</ul>

</nav>

</header>

<main>

<h1>Good Old Fashioned Pancakes</h1>

<img

src="https://images.unsplash.com/photo-1575853121743-60c24f0a7502?ixid=MXwxMjA3fDB8MHxzZWFyY2h8MXx8cGFuY2FrZXxlbnwwfHwwfA%3D%3D&ixlib=rb-1.2.1&auto=format&fit=crop&w=700&q=60"

alt="pancake"

width="250"

/>

<h2 id="ingredients">Ingredients</h2>

<ol>

<li><input type="checkbox" /> 1 ½ cups all-purpose flour</li>

<li><input type="checkbox" /> 3 ½ teaspoons baking powder</li>

<li><input type="checkbox" /> 1 teaspoon salt</li>

<li><input type="checkbox" /> 1 tablespoon white sugar</li>

<li><input type="checkbox" /> 1 ¼ cups milk</li>

<li><input type="checkbox" /> 1 egg</li>

</ol>

<h2 id="steps">Steps</h2>

<h4>Step 1</h4>

<p>

In a large bowl, sift together the flour, baking powder, salt and sugar.

Make a well in the center and pour in the milk, egg and melted butter;

mix until smooth.

</p>

<h4>Step 2</h4>

<p>

Heat a lightly oiled griddle or frying pan over medium-high heat. Pour

or scoop the batter onto the griddle, using approximately 1/4 cup for

each pancake. Brown on both sides and serve hot.

</p>

</main>

<hr />

<footer>

<h6 id="subscribe">Subscribe</h6>

<form onsubmit="alert('Subscribed')">

<input type="text" placeholder="Enter Email Address" />

<button>Submit</button>

</form>

<br />

<div>&copy; dakota kelly at Allrecipe.com</div>

</footer>

</body>

</html>

## Conclusion

You can build a simple website with just HTML. But to be able to build beautiful and functional websites, you need to study CSS and JavaScript.

# HTML5 Semantic Elements

<https://www.freecodecamp.org/news/semantic-html5-elements/#:~:text=Semantic%20HTML%20elements%20are%20those,content%20that%20is%20inside%20them>.

Semantic HTML elements are those that clearly describe their meaning in a human- and machine-readable way.

Elements such as <header>, <footer> and <article> are all considered semantic because they accurately describe the purpose of the element and the type of content that is inside them.

## **What are Semantic Elements?**

HTML was originally created as a markup language to describe documents on the early internet. As the internet grew and was adopted by more people, its needs changed.

Where the internet was originally intended for sharing scientific documents, now people wanted to share other things as well. Very quickly, people started wanting to make the web look nicer.

Because the web was not initially built to be designed, programmers used different hacks to get things laid out in different ways. Rather than using the <table></table> to describe information using a table, programmers would use them to position other elements on a page.

As the use of visually designed layouts progressed, programmers started to use a generic “non-semantic” tag like <div>. They would often give these elements a class or id attribute to describe their purpose. For example, instead of <header> this was often written as <div class="header">.

As HTML5 is still relatively new, this use of non-semantic elements is still very common on websites today.

### List of new semantic elements

The semantic elements added in HTML5 are:

* <article>
* <aside>
* <details>
* <figcaption>
* <figure>
* <footer>
* <header>
* <main>
* <mark>
* <nav>
* <section>
* <summary>
* <time>

Elements such as <header>, <nav>, <section>, <article>, <aside>, and <footer> act more or less like <div> elements. They group other elements together into page sections. However where a <div> tag could contain any type of information, it is easy to identify what sort of information would go in a semantic <header> region.

## Why use semantic elements?

To look at the benefits of semantic elements, here are two pieces of HTML code. This first block of code uses semantic elements:

<header></header>

<section>

<article>

<figure>

<img>

<figcaption></figcaption>

</figure>

</article>

</section>

<footer></footer>

Whilst this second block of code uses non-semantic elements:

<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>

First, it is much **easier to read**. This is probably the first thing you will notice when looking at the first block of code using semantic elements. This is a small example, but as a programmer you can be reading through hundreds or thousands of lines of code. The easier it is to read and understand that code, the easier it makes your job.

It has **greater accessibility**. You are not the only one that finds semantic elements easier to understand. Search engines and assistive technologies (like screen readers for users with a sight impairment) are also able to better understand the context and content of your website, meaning a better experience for your users.

Overall, semantic elements also lead to more **consistent code**. When creating a header using non-semantic elements, different programmers might write this as <div class="header">, <div id="header">, <div class="head">, or simply <div>. There are so many ways that you can create a header element, and they all depend on the personal preference of the programmer. By creating a standard semantic element, it makes it easier for everyone.

Since October 2014, HTML4 got upgraded to HTML5, along with some new “semantic” elements. To this day, some of us might still be confused as to why so many different elements that doesn’t seem to show any major changes.

## <section>**and**<article>

“What’s the difference?”, you may ask. Both these elements are used for sectioning a content, and yes, they can definitely be used interchangeably. It’s a matter of in which situation. HTML4 offered only one type of container element, which is <div>. While this is still used in HTML5, HTML5 provided us with <section> and <article> in a way to replace <div>.

The <section> and <article> elements are conceptually similar and interchangeable. To decide which of these you should choose, take note of the following:

1. An article is intended to be independently distributable or reusable.
2. A section is a thematic grouping of content.

<section>

<p>Top Stories</p>

<section>

<p>News</p>

<article>Story 1</article>

<article>Story 2</article>

<article>Story 3</article>

</section>

<section>

<p>Sport</p>

<article>Story 1</article>

<article>Story 2</article>

<article>Story 3</article>

</section>

</section>

## <header>**and**<hgroup>

The <header> element is generally found at the top of a document, a section, or an article and usually contains the main heading and some navigation and search tools.

<header>

<h1>Company A</h1>

<ul>

<li><a href="/home">Home</a></li>

<li><a href="/about">About</a></li>

<li><a href="/contact">Contact us</a></li>

</ul>

<form target="/search">

<input name="q" type="search" />

<input type="submit" />

</form>

</header>

The <hgroup> element should be used where you want a main heading with one or more subheadings.

<hgroup>

<h1>Heading 1</h1>

<h2>Subheading 1</h2>

<h2>Subheading 2</h2>

</hgroup>

REMEMBER, that the <header> element can contain any content, but the <hgroup> element can only contain other headers, that is <h1> to <h6> and including <hgroup>.

## <aside>

The <aside> element is intended for content that is not part of the flow of the text in which it appears, however still related in some way. This of <aside> as a sidebar to your main content.

<aside>

<p>This is a sidebar, for example a terminology definition or a short background to a historical figure.</p>

</aside>

Before HTML5, our menus were created with <ul>’s and <li>’s. Now, together with these, we can separate our menu items with a <nav>, for navigation between your pages. You can have any number of <nav> elements on a page, for example, its common to have global navigation across the top (in the <header>) and local navigation in a sidebar (in an <aside> element).

<nav>

<ul>

<li><a href="/home">Home</a></li>

<li><a href="/about">About</a></li>

<li><a href="/contact">Contact us</a></li>

</ul>

</nav>

## <footer>

If there is a <header> there must be a <footer>. A <footer> is generally found at the bottom of a document, a section, or an article. Just like the <header> the content is generally metainformation, such as author details, legal information, and/or links to related information. It is also valid to include <section> elements within a footer.

<footer>&copy;Company A</footer>

## <small>

The <small> element often appears within a <footer> or <aside> element which would usually contain copyright information or legal disclaimers, and other such fine print. However, this is not intended to make the text smaller. It is just describing its content, not prescribing presentation.

<footer><small>&copy;Company A</small> Date</footer>

## <time>

The <time> element allows an unambiguous ISO 8601 date to be attached to a human-readable version of that date.

<time datetime="2017-10-31T11:21:00+02:00">Tuesday, 31 October 2017</time>

Why bother with <time>? While humans can read time that can disambiguate through context in the normal way, the computers can read the ISO 8601 date and see the date, time, and the time zone.

## <figure> and <figcaption>

<figure> is for wrapping your image content around it, and <figcaption> is to caption your image.

<figure>

<img src="https://en.wikipedia.org/wiki/File:Shadow\_of\_Mordor\_cover\_art.jpg" alt="Shadow of Mordor" />

<figcaption>Cover art for Middle-earth: Shadow of Mordor</figcaption>

</figure>

# HTML Metadata

<https://www.freecodecamp.org/news/meta-tag-in-html-what-is-metadata-and-meta-description-example/>

## How to set up an HTML project

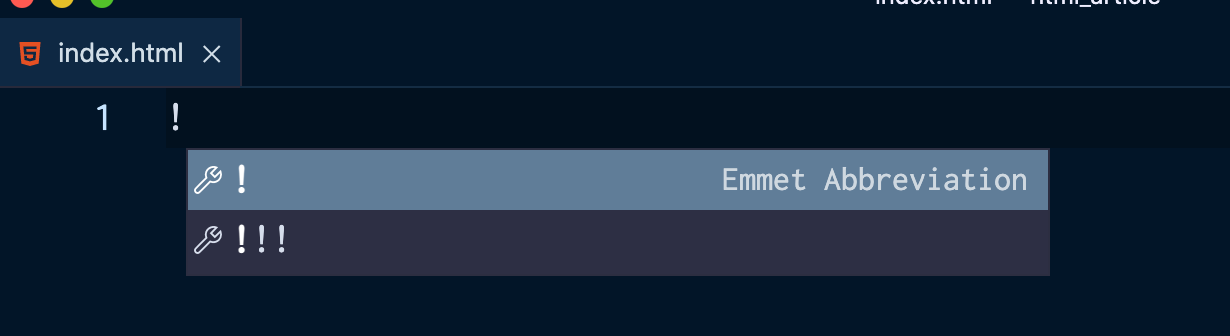
When setting up new HTML projects, you'll find that you have to include the same few tags every single time.

These tags are essential, and you'll need them to get your HTML site up and running properly, following best practices.

Some code editors offer shortcuts to automatically fill out and enter the tags that you use in every new HTML project. This can save you considerable time.

In the [Visual Studio Code editor](https://code.visualstudio.com/download), you can do this in the following way:

1. Make sure you've created a file ending in .html - here you'll write all of your HTML code.
2. Inside the empty file type an exclamation mark, !.



1. Click on the exclamation mark with the mention that the following is an Emmet Abbreviation.

[Emmet](https://emmet.io/) is a plugin for code editors that's built into Visual Studio Code by default, and it helps you optimise your HTML workflow.

You'll then see the following code filled out:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

</body>

</html>

When viewing the .html file in the browser of your choice, you'll see just an empty page.

Le'ts zoom in to the following section of the code that was created:

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

What are these meta tags exactly? Why are they there and what purpose do they serve when creating a webpage?

This article will focus on explaining the basics of meta tags and why they are used in HTML documents.

## What are meta tags in HTML?

meta tags live within the head tag of the HTML document.

The head tag is used for configurating the HTML file.

You use the head tag to add a title to the webpage, link to a CSS stylesheet, and define more information about the HTML document.

meta tags represent metadata. They are essentially used for defining and describing data about data, and are used to add extra information to the data inside the webpage.

There are many meta tags. Some of them help improve the SEO (Search Engine Optimisation) of your website, making sure that the content of your site is relevant to what people are searching for.

### How to define the character set of a website

<meta charset="UTF-8"> defines the character set that will be used in the site.

UTF-8, which stands for 8-bit Unicode Transformation Format, is the standard character encoding used with the latest version of HTML, which is HTML5.

This line should be included in every single webpage created, as it ensures that every character from every language in the world is displayed properly in every browser.

By using the universal UTF-8 as the character set, characters from non-latin languages will not be distorted.

The Google Chrome browser has automatically set the encoding to UTF-8, so you won't have to worry about that when designing for this browser. But you still need to include <meta charset="UTF-8"> in every HTML file in case this feature is not supported by other browsers.

For example, look at what happens in the Safari browser when this line is not added and I write a heading in a non-latin language, such as Greek:

<!DOCTYPE html>

<html lang="en">

<head>

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

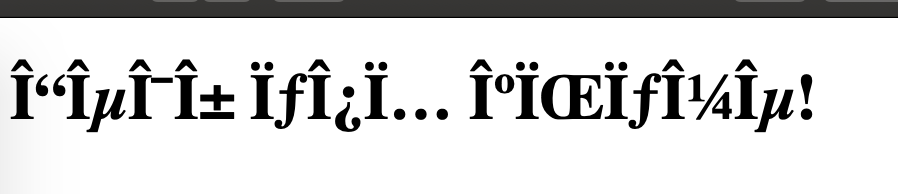
</head>

<body>

<h1>Γεία σου κόσμε!</h1> <!-- Hello world! -->

</body>

</html>



When the HTML document is viewed in the browser, all the characters are distorted.

### How to let Microsoft's Internet Explorer know which rendering view to use

You use the http-equiv="X-UA-Compatible" content="IE=edge" meta tag to choose and define the version of Internet Explorer in which the web page will be renedered.

Always choose the latest one, which is IE=edge.

There are many versions of Microsoft's browser. In the past the different advances caused headaches to web designers and web developers alike, who worked on making sure websites were usable on legacy browsers.

This tag will ensure that the website will not be rendered as an older version of Internet Explorer, which tend to be buggy.

### How to adjust viewport settings

Nowadays, it is important that all sites look good on all devices, especially mobile phones.

So, you need to include the meta name="viewport" content="width=device-width, initial-scale=1.0" tag in every HTML file.

viewport refers to how the site is displayed on different screen sizes, and how much visual area a user has available.

Each device has a different viewport. For example, mobile devices have a smaller one and desktop computers have a larger one.

content="width=device-width is the first step to making sure that websites look good on mobile devices.

It prevents a site that is viewed from a mobile device from looking like it would on a laptop – that is small and far away zoomed out.

This ensures that the HTML will adjust to the width of the device's screen.

initial-scale=1.0 sets how the webpage scales,and sets the initial zoom when the page is first loaded by the browser.

## Additional **meta** tags to add to your HTML project

### How to add a description of your webpage

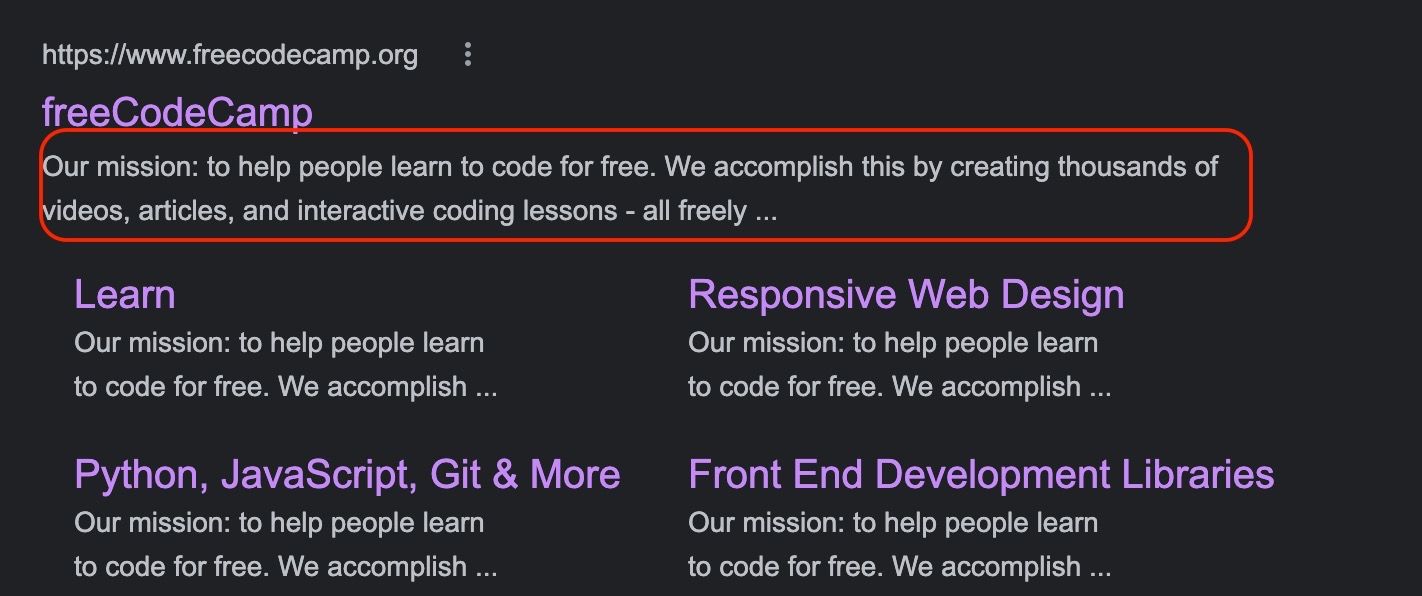
Using a meta description tag for your page helps search engines figure out and rank your website against other websites. It's used primaraly for SEO (Search Engine Optimization) purposes.

The meta description tag is used to explain in a brief and concise way what your website is about.

A meta description tag could look something like this:

<meta name="description" content="Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public.">

You use the name and content attributes, with the text value passed to content showing up in the search results:



#### Things to consider when writing a description of your website

* Make sure to keep the description of your website short and not over 160 characters.
* Include useful keywords and key-phrases that people tend to use often when searching for the services that your website provides.
* Explain clearly what your website does and the mission behind it. It is important to get across what sets you apart and the value you provide.
* Be consistent with the voice and tone of your brand.
* Most importantly, stick to describing the content your website actually provides. Don't try and trick your readers by only aiming to appear high in searches and rankings.

### How to add the name of the website's author

Another useful meta element to include is the author's name.

This could look like the following:

<meta name="author" content="Quincy Larson">

It can be helpful to know who authored the page.

This info shares who created and built the website, who authored the content, or to whom the copyright belongs.

## Conclusion

To summarize, all HTML documents need to include at least the following three meta tags:

* <meta charset="UTF-8">, to specify the character set.
* <meta name="description>", to add a clear description of the site and the services the site provides to readers/customers.
* <meta name="viewport>, which is the first step sites need to take to be usable on a variety of screen devices.

# Web Accessibility

<https://blog.hubspot.com/website/html-accessibility>

<https://html-css-js.com/>

Accessible design is that which accommodates everyone, which special attention to those with disabilities and limitations. We see accessibility baked into the world around us, from closed captioning to [curb cuts](https://99percentinvisible.org/episode/curb-cuts/).

When it comes to [accessibility on the web](https://blog.hubspot.com/website/web-accessibility), however, we have a lot more work to do. [According to a recent survey of screen reader users](https://webaim.org/projects/screenreadersurvey8/), 60% of respondents believed that overall web accessibility had either remained the same or had worsened since the previous year.

The good news is that, with a bit of extra care taken on the back-end, any website can be made accessible to users of assistive technologies like [screen readers](https://blog.hubspot.com/website/screen-reader-accessibility). By investing in web accessibility, not only do you ensure a positive user experience for everyone — you also show your visitors that you care about equal access to your content. And it all starts with taking a look at your site’s underlying HTML.

In this guide, you’ll learn the basics of how to build a web page that works with assistive technologies — we’ll review six aspects of HTML that make this possible. For those unfamiliar with web development, we recommend first reviewing our [introduction to HTML, CSS, and JavaScript](https://blog.hubspot.com/marketing/web-design-html-css-javascript), as well as our [full guide to HTML](https://blog.hubspot.com/website/html), before continuing.

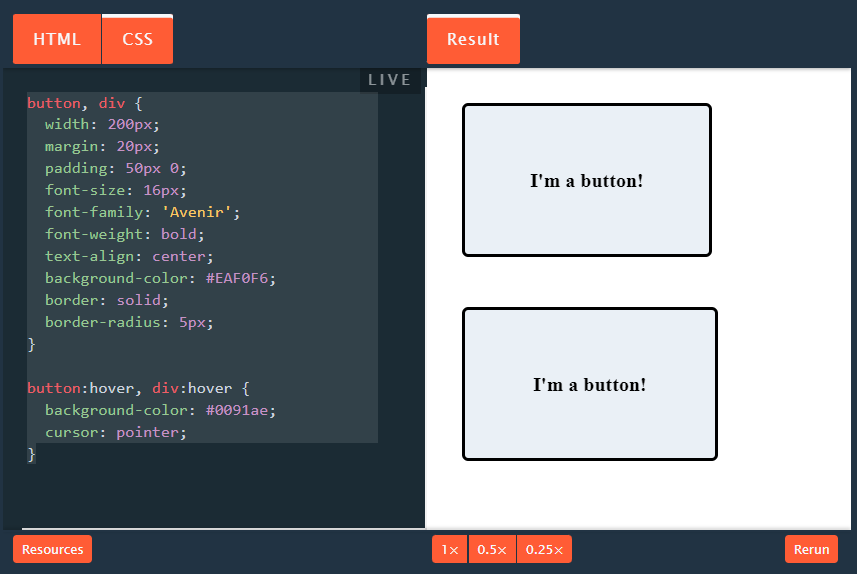
## Semantic HTML

HTML — as well as its companion style language, CSS — is flexible. If you want to build a certain type of page, there are probably multiple ways you could go about writing it with HTML and styling it with CSS. However, not all HTML elements are made equal when it comes to accessibility.

When writing web pages, the single best way to make them accessible is to use semantic HTML. Semantic HTML is HTML code that says what it does — in other words, the tag itself conveys the purpose of the element. Semantically rich elements include **<button>**, **<form>**, **<header>**, **<footer>**, **<nav>**, and the headings **<h1>**, **<h2>**, etc.

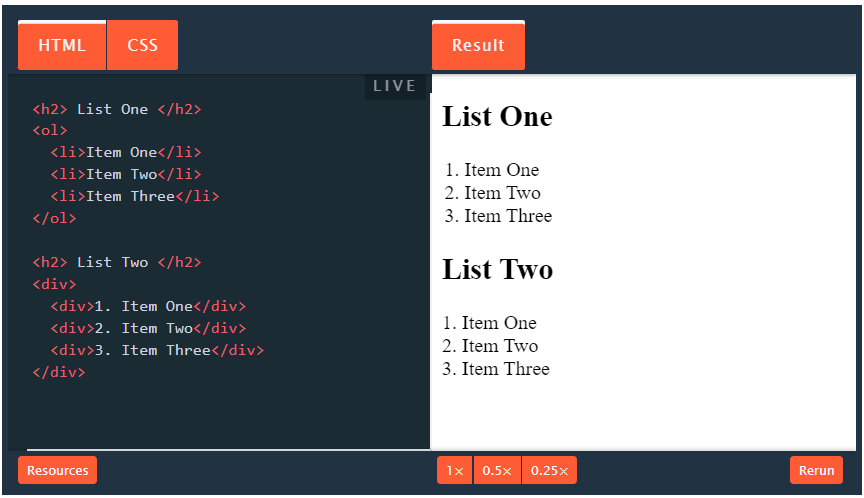
Consider the example below. Here, we have two button elements that, at least to the human eye, look the same:

|  |  |
| --- | --- |
| **HTML** | **CSS** |
| <button>I'm a button!</button>  <div>I'm a button!</div> | button, div {  width: 200px;  margin: 20px;  padding: 50px 0;  font-size: 16px;  font-family: 'Avenir';  font-weight: bold;  text-align: center;  background-color: #EAF0F6;  border: solid;  border-radius: 5px;  }  button:hover, div:hover {  background-color: #0091ae;  cursor: pointer;  } |



But, if you reveal the underlying HTML, you’ll see that the first element is a **<button>** element, while the second is a [**<div>**](https://blog.hubspot.com/website/what-div-in-html) element. The former is semantic, while the latter is not.

Another example: Below, we have two lists. However, the first uses the semantic **<ol>** (ordered list) tag, while the second is formatted with generic tags:



While the difference between semantic and non-semantic tags might not seem important to sighted users, it matters significantly to those using screen readers for several reasons:

First, semantic HTML tells screen reader users exactly what they’re viewing. The semantic **<button>** element tells the screen reader that this element is meant to be clicked to perform some action, and the **<ol>** element says that the elements it contains are part of a numbered list. Since **<div>** is not semantically rich, it does not convey this information — all the user knows is that the element in question is some generic block-level … thing.

Second, semantic elements come with their own keyboard accessibility built-in, with no extra work required on your part. The **<button>** element, for example, lets users “click” the element with the Enter/Return key, and focus on it with the Tab key. This isn’t the case with **<div>**. When you use semantic elements, you enable complimentary accessible functionality.

Lastly, many screen readers help users navigate a page by letting them skip between tags of the same name (i.e., H2s), or aggregate all of the same tags for an easier way to scan the contents of a page. So, it’s clear why using mostly **<div>** tags would negate this feature.

To sum up, avoid creating specific interaction elements with the generic **<div>** tag and its inline sibling, **<span>**, if you can. While these tags are often useful for layouts, always choose the native semantic HTML element when possible, for better accessibility.

## Image Alt Text

If you’re a marketer, you’ve likely heard of alternative text, or “alt text,” in the context of SEO. Alt text tells search engine crawlers the contents of your images, and is an opportunity to provide more information to rank higher in SERPs.

However, the original purpose of alt text was to add context for those using screen readers and for those who cannot see your images for any reason (e.g., low vision, a poor connection, or a broken source link). When a screen reader encounters alt text, it simply reads it to the user. Browsers can also be set to display alt text onscreen in lieu of the image.

Image alt text isn’t hard to implement. In the code, it’s as easy as placing the **alt** attribute inside the **<img>** tag like so:

<img src="cute-dog.jpg" alt="a small, brown sitting in a field of dandelions with a chew toy in its mouth">

If an image lacks the **alt** attribute, a screen reader will either read the image filename, **title** attribute if one exists, or skip over the image altogether. All of these outcomes prevent users from fully accessing your content, so it’s important to add [concise, clear alt text](https://blog.hubspot.com/marketing/image-alt-text) to all non-decorative **<img>** tags on your page. Your alt text should directly describe the contents of the image. It’s okay to include a keyword or two, but avoid keyword stuffing, a practice which impedes accessibility.

For decorative images inside your HTML file, those which add visual style but are not essential to understand the page content, include the **alt** attribute but leave the value empty. This tells the screen reader that an image exists but is not important to the reader. It prevents the screen reader from reading the image file name or **title** attribute.

<img src="background-image.jpg" alt="">

Note that decorative images may be better implemented as a [CSS background image](https://blog.hubspot.com/website/insert-image-in-html) rather than part of your HTML.

## Well-Structured HTML

When sighted users land on a web page, their eyes dart around the page in search of their desired content. A screen reader, however, processes pages in a linear manner — it starts at the top of the HTML and reads down through each page element one-by-one.

To help screen reader users better understand your pages, it helps to structure your content in a useful way.

This means two things: First, use semantic HTML tags to segment different regions of your page. For example:

* Use heading and subheading tags (**<h1>**, **<h2>**, **<h3>**, etc.) to designate text areas by order of importance. **<h1>** is the most visible heading and should only be used once.
* Indicate separate paragraphs inside separate **<p>** tags.
* Indicate numbered lists and non-numbered lists with **<ol>** and **<ul>** respectively. Nested listed items use **<li>** tags.
* Use **<header>** and **<footer>** tags for your header and footer regions.
* Place any navigational elements inside the **<nav>** tag.
* Place the main page content inside the **<main>** tag. Use this tag only once per page. Main page content is any content that isn’t repeated across pages on your site, so it excludes things like headers, footers, and navigation.

Second, order these tags logically. Usually, visitors expect your header and navigation to come first, followed by **<main>** content, followed by your footer. Inside **<main>**, place your headers in descending order, and do not skip header levels (i.e. don’t place an **<h4>** element directly under an **<h2>** element without an **<h3>** in between).

Well-structured semantic HTML helps screen reader users better understand the layout of your page. It also lets them preview sections and skip between different content regions, similar to how sighted readers skim web pages. This HTML structure is much more navigable than a clump of generic **<div>** and **<span>** tags.

## The HTML lang Attribute

Here’s a quick win for writing more accessible HTML — every page on your website should include the **lang** (language) attribute placed inside the opening **<html>** tag. This identifies the primary language of the page to the browser, search engines, translation software, and screen readers.

Add your **lang** attribute like so:

<html lang=“en”>

<!-- the rest of the page content, in English -->

</html>

If you have an instance of a foreign word on your page, you can place the **lang** attribute inside any other tag, such as **<p>** or a **<span>**, to indicate a temporary language change:

<html lang=”en”>

    <body>

     <p>The language of this document is English, as indicated in the html tag above.</p>

     <p>We can also use the lang attribute to indicate a foreign word, like <span lang=“fr”>salut</span> or <span lang=“de”>hallo</span>.

</p>

     <p lang=“es”>Este párrafo está en español.</p>

   </body>

</html>

For screen readers specifically, the **lang** attribute tells the text-to-speech function how to pronounce the words on a page. If the wrong **lang** value is used, visitors will hear your page read in, at best, a heavy accent. This is why it’s a good idea to check that all of your pages are set to the right language.

## Descriptive Hyperlinks

When designing for better accessibility, we want to inform users what to expect before they take a particular action on a page. The most common action a user can take is clicking a link to a different page — before a user does this, they should know where they are going from the link text alone, without any additional context.

To achieve this, write link text that adequately describes where it leads and is unique from other link text. Consider the following two sentences:

1. [**Click here**](https://blog.hubspot.com/website/web-accessibility) to learn more about web accessibility.
2. To learn more about accessible websites, [**see our Ultimate Guide to Web Accessibility**](https://blog.hubspot.com/website/web-accessibility).

Though people reading these sentences will know what to expect from both of these links, the problem is that a screen reader user might pull a list of all links on a page and select one from this list. Someone reading the link text from example 1 in isolation won’t know where it leads. Plus, if there are multiple links that read “Click here,” it’s not apparent if or how these links are different.

On the other hand, it’s clear from the text in example 2 that we’re linking to our web accessibility guide.

## ARIA Roles

The Accessible Rich Internet Applications suite of web standards, shortened to ARIA, is a set of standards to make user interface controls more accessible for those using assistive technologies.

The ARIA documentation outlines how to use HTML attributes called [ARIA roles](https://www.w3.org/TR/wai-aria-1.1/#usage_intro), which specify the function of a particular interactive element or landmark element for screen reader users who may not be able to visually grasp the purpose of an element. ARIA roles are best used on **<div>** and **<span>** tags.

There are [ARIA roles for most page elements](https://www.w3.org/TR/wai-aria-1.1/#roles). For example, to indicate that a list contains navigational links, you can use the ARIA role **navigation**:

<div role=”navigation”>

    <ul>

       <li>Product</li>

       <li>Solutions</li>

       <li>About Us</li>

       <li>Blog</li>

   </ul>

</div>

ARIA is a complex accessibility topic in and of itself — to fully adhere to ARIA guidelines, we recommend consulting the [full documentation](https://www.w3.org/TR/wai-aria-1.1/) or consulting with an accessibility specialist.

We can now use HTML5 elements to convey that same document structure to screen readers and voice recognition software, to allow users to easily navigate to those areas of the page. There are also special ARIA roles called landmark roles that serve the same purpose.

These are some (but not all) of the elements that you can use. This is just a basic overview, so visit the links to get more detail on how they should be used.

|  |  |  |
| --- | --- | --- |
| **HTML5** | **ARIA** | **Description** |
| [<header>](http://html5doctor.com/the-header-element/) | role=”banner” | Introduction to a page or section. Can contain a heading (H1-H6), site logo, navigation. |
| [<nav>](http://html5doctor.com/nav-element/) | role=”navigation” | Can be used for various types of navigation such as site navigation, subnavigation, breadcrumbs, previous/next links. |
| [<footer>](http://html5doctor.com/the-footer-element-update/) | role=”contentinfo” | Describes the page or a section of the page. A page’s footer may contain author name, copyright info, privacy policy, etc. |
| [<aside>](http://html5doctor.com/aside-revisited/) | role=”complementary” | Information that is tangentially related to the main page content but can be read separately. Visually you might see this as a sidebar. |
| [<article>](http://html5doctor.com/the-article-element/) | no equivalent | Independent item such as a blog post, article, etc. Think of it as something that could be independently picked up and moved around, such as blog posts in an RSS feed. |

## Accessibility Starts With Your HTML

After implementing the measures above, try running your page through an [accessibility testing tool](https://blog.hubspot.com/website/web-accessibility-testing-tools). Here’s [one accessibility checker](https://www.webaccessibility.com/) endorsed by the Web Accessibility Initiative that grades your website on a zero-to-100 scale and lists any accessibility violations that you can address.

Like many aspects of web design, accessibility is best considered from the start of the building process, rather than an afterthought. Implementing accessible best practices from the start will save you from having to revamp your pages later on.

However, at their most basic level, web pages are just text documents that can be reconfigured as needed. With some extra tweaking and several accessibility checks, you’ll quickly improve the browsing experience for your visitors who need it. After that, move on to other [content-related changes that help your visitors](https://blog.hubspot.com/website/web-accessibility-guidelines).

Everyone deserves equal access to public information online. To make the web a better place, start with your own site.