
HTML5 TRAINING

Guide By Skyline Ict Consult Ltd
Olisa Macaulay



STUDENT NAME: Peter Enosakhare Okhuakhu

STUDENT PASS CODE: BFB-0018

This HTML5 Course Covers the following.



HTML Introduction	01
HTML Editors	02
HTML Basic	03
HTML Elements	04
HTML Attributes	05
HTML Headings	06
HTML Paragraphs	07
HTML Styles	08
HTML Formatting	09
HTML Quotations	10

HTML Comments	11
HTML Colors	13
HTML CSS	14
HTML Links	15
HTML Images	16
HTML Favicon	17
HTML Tables	18
HTML Lists	20
HTML Block & Inline	21
HTML Classes	22

This HTML5 Course Covers the following.(2)

HTML Id	22
HTML Iframes	23
HTML File Paths	24
HTML Head	25
HTML Layout	26
HTML Responsive	27
HTML Semantics	30
HTML Forms	31

HTML Form Attribute	32
HTML Form Elements	33
HTML Input Types	34
HTML Input Attributes	35
Input Form Attributes	36
HTML Video	37
HTML Audio	38
HTML YouTube	39

- What you should expect from me

I take my courses very seriously but at the same time I try to make it fun since I know how difficult learning from an instructor This course is fun, and when you need some energy to keep going, you will get it from me.



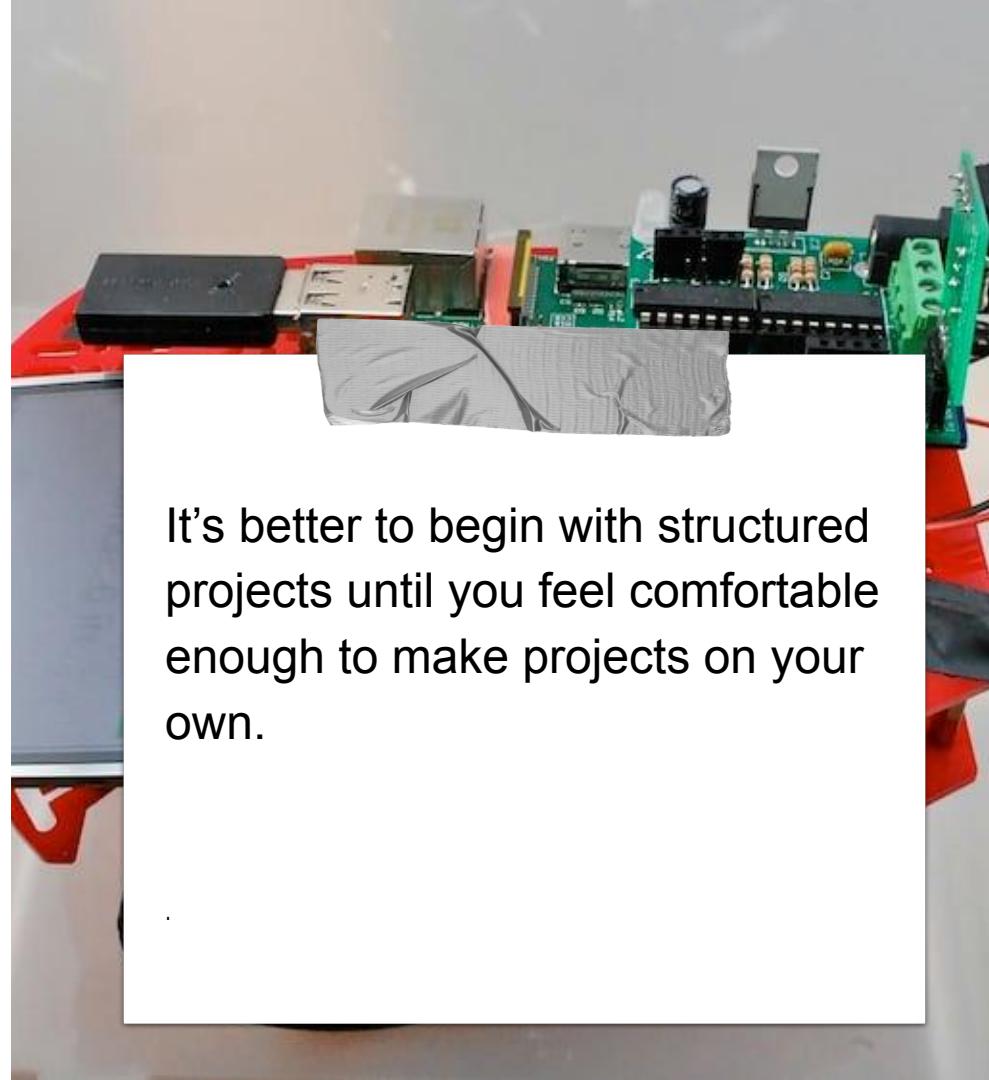
-

Practice, practice and more practice. Every section inside this course has a practice lecture and exercise at the end, reinforcing everything with went over in the lectures.



What's the lesson here? You need to find what motivates you and get excited about it! To get started, find one or two areas that interest you

Once you've learned the basic HTML syntax, start doing projects. Applying your knowledge right away will help you remember everything you've learned.



It's better to begin with structured projects until you feel comfortable enough to make projects on your own.

FRONT-END DEVELOPER

A front end developer, is a professional responsible for the design and implementation of the website interface.

Have you ever looked at your favorite website and wondered why it looked like that, how the buttons worked, or thought, “I wonder how complicated that is?” or, I wish I could do that? While web design determines the way a website looks, front end development is how that design actually gets implemented on the web.

A front end web developer is a **software engineer** who implements web designs through coding languages like **HTML, CSS, and JavaScript**. If you head to any site, you can see the work of a front end developer in the navigation, layouts (including this article), and in the way that a site looks different on your phone (thanks to mobile-first or responsive design).

Key Front End Developers Skills



Front end web developers use three primary coding languages to code the website

HTML,CSS, JavaScript

The code front end developers write runs inside the user's web browser (known as client-side,

as opposed to a back end developer, whose code runs server-side using open source runtime environments like **Django** or with programming languages like Python). **Full stack developers** are comfortable programming with **both front end and back end languages**.

A back end developer is like the engineer who designs and creates the systems that make a city work (electricity, water and sewer, zoning, etc.), while the front end developer is the one who lays out the streets and makes sure everything is connected properly so people can live their lives.

HTML



CSS



JS



Before we begin, install:



1. Web Browser
(lets us view websites)



2. Code Editor
(helps us write code)



VS Code

Structuring the web Section: 1

To build websites, you should know about HTML — the fundamental technology used to define the structure of a webpage.

HTML is used to specify whether your web content should be recognized as a paragraph, list, heading, link, image, multimedia player, form, or one of many other available elements

After learning HTML, you can then move on to learning about more advanced topics such as CSS, and how to use it to style HTML (for example, alter your text size and fonts used, add borders and drop shadows, layout your page with multiple columns, add animations and other visual effects).

JavaScript, and how to use it to add dynamic functionality to web pages (for example, find your location and plot it on a map, make UI elements appear/disappear when you toggle a button, save users' data locally on their computers, and much more).



What is HTML?

HTML is the language in which most websites are written. HTML is used to create pages and make them functional.

The code used to make them visually appealing is known as **CSS** and we shall focus on this in a later class

For now, we will focus on teaching you how to build rather than design.



The History of HTML

HTML was first created by **Tim Berners-Lee, Robert Cailliau**, and others starting in 1989. It stands for Hyper Text Markup Language.

Hypertext means that the document contains links that allow the reader to jump to other places in the document or to another document altogether. The latest version is known as HTML5.

A Markup Language is a way that computers speak to each other to control how text is processed and presented. To do this HTML uses two things: tags and attributes.

Installing Visual Studio Code on Windows

Visual Studio Code is the most popular code editor and the IDEs provided by Microsoft for writing different programs and languages. It allows the users to develop new code bases for their applications and allow them to successfully optimize them and debug them properly.

It is a very user-friendly code editor and it is supported on all the different types of operating systems like Windows, macOS, and Linux.

It has the support for all the languages like HTML, CSS, Java, Python, JavaScript, React, Node JS, etc.

Follow the below steps to install **Visual Studio Code** on Windows:

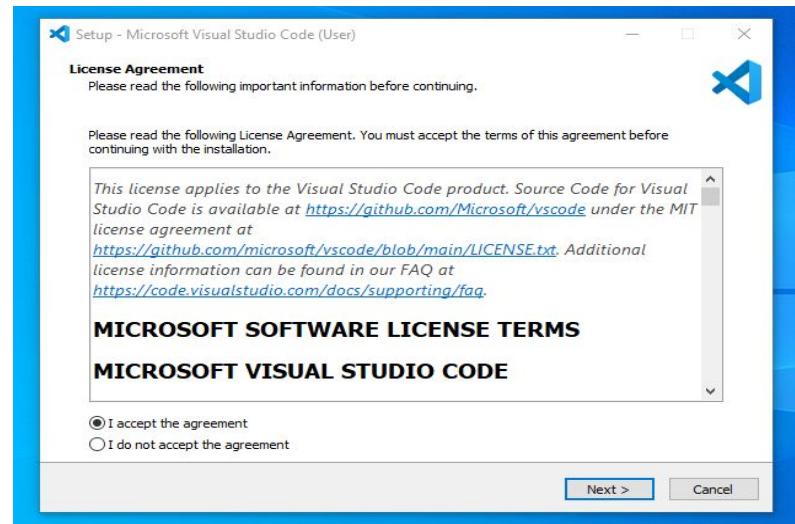
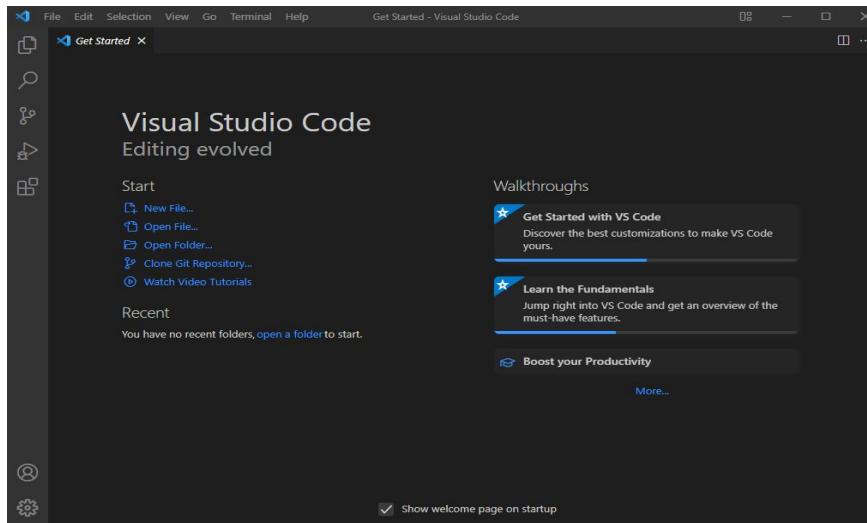
Step 1: Visit the [official website](#) of the **Visual Studio Code** using any web browser like Google Chrome, Microsoft Edge, etc.



Step 2: Click on the installer icon to start the installation process of the Visual Studio Code.

Step 3: After the Installer opens, it will ask you for accepting the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.

Step 4: Choose the location data for running the Visual Studio Code. It will then ask you for browsing the location. Then click on Next button.



A Simple HTML Document

Example

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

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

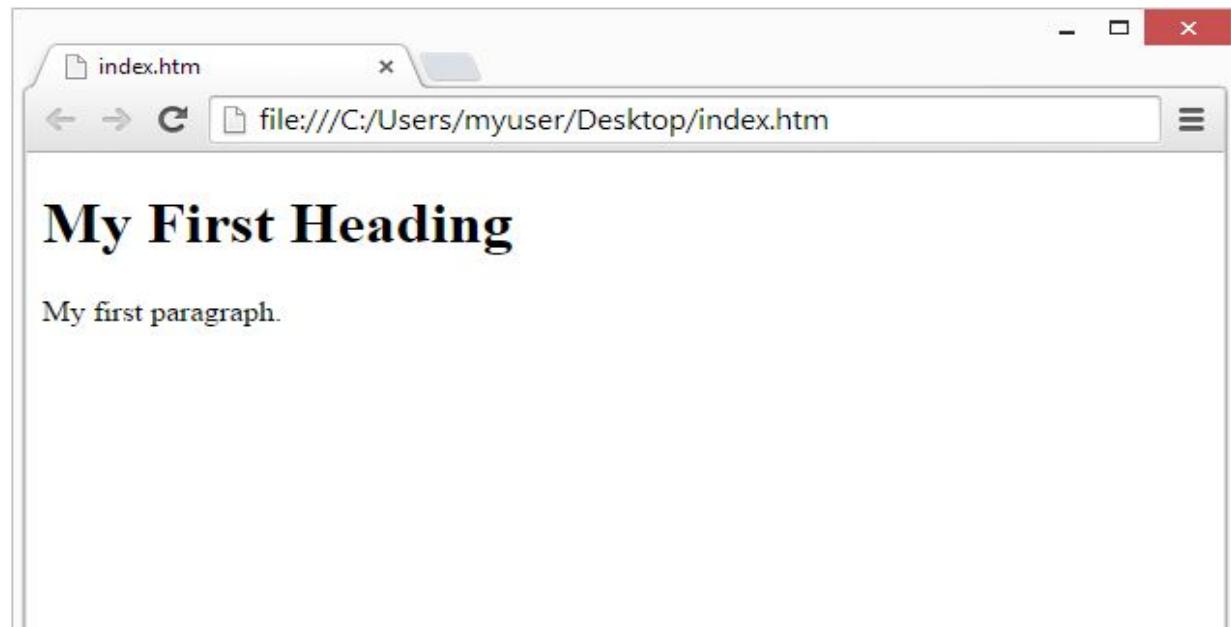
</body>
</html>
```



Web Browsers

The purpose of a web browser (Chrome, Edge, Firefox, Safari) is to read HTML documents and display them correctly.

A browser does not display the HTML tags, but uses them to determine how to display the document:



Example Explained

- The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the HTML page
- The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

HTML Editors Section 2.

HTML text editors are used to create and modify web pages. HTML codes can be written in any text editors including the **notepad**. One just needs to write HTML in any text editor and save the file with an extension “.html” or “.htm”. Some of the popular HTML text editors are given below:

Notepad

VS CODE

Notepad++

Sublime Text 3

Atom



A simple text editor is all you need to learn HTML.

Learn HTML Using Notepad or TextEdit

Web pages can be created and modified by using professional HTML editors.

However, for learning HTML we recommend a simple text editor like Notepad (PC) or TextEdit (Mac).

We believe that using a simple text editor is a good way to learn HTML.

Follow the steps below to create your first web page with Notepad or TextEdit.



Step 1: Open Notepad (PC)

Windows 8 or later:

Open the **Start Screen** (the window symbol at the bottom left on your screen). Type **Notepad**.

Windows 7 or earlier:

Open **Start > Programs > Accessories > Notepad**

Step 1: OpenTextEdit (Mac)

Open **Finder > Applications > TextEdit**

Also change some preferences to get the application to save files correctly. In **Preferences > Format >** choose "**Plain Text**"

Then under "Open and Save", check the box that says "Display HTML files as HTML code instead of formatted text".

Then open a new document to place the code.

Step 2: Write Some HTML

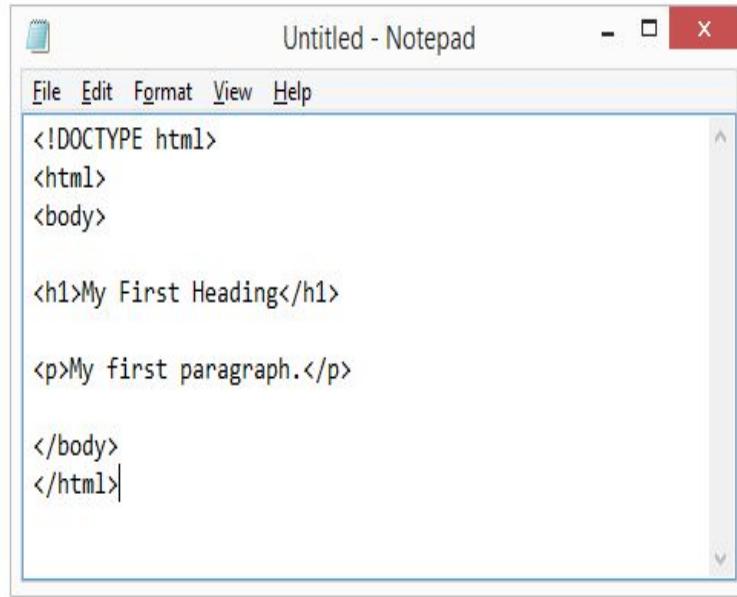
Write or copy the following HTML code into Notepad:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```



A screenshot of the Windows Notepad application titled "Untitled - Notepad". The window contains the following HTML code:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

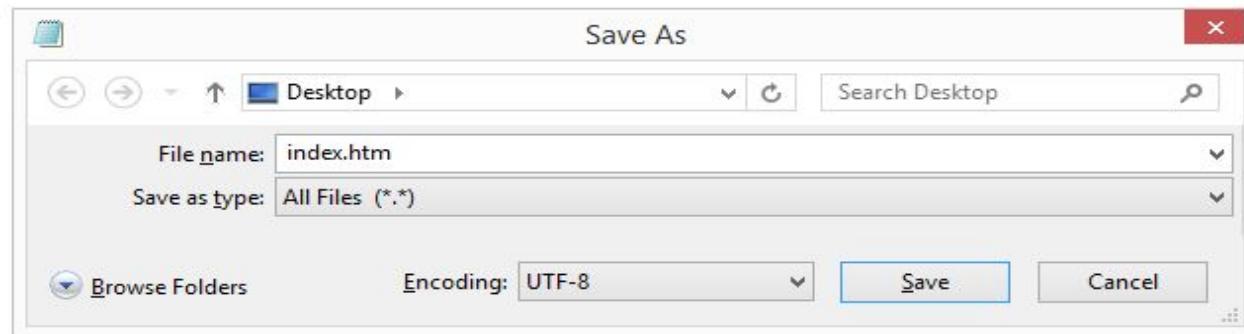
</body>
</html>
```



Step 3: Save the HTML Page

Save the file on your computer. Select **File > Save as** in the Notepad menu.

Name the file "**index.htm**" and set the encoding to **UTF-8** (which is the preferred encoding for HTML files).

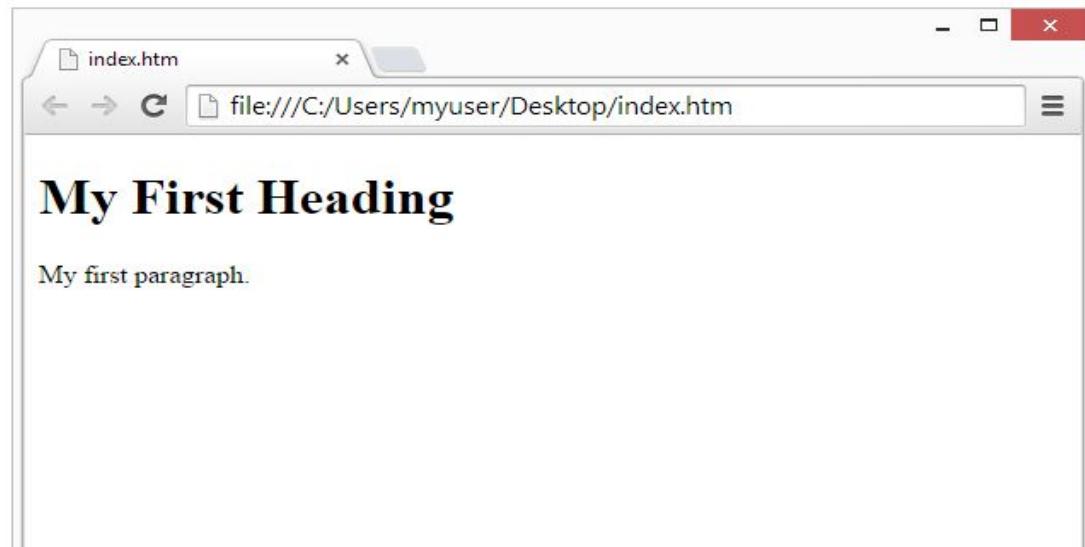


Tip: You can use either .htm or .html as file extension. There is no difference; it is up to you.

Step 4: View the HTML Page in Your Browser

Open the saved HTML file in your favorite browser (double click on the file, or right-click - and choose "Open with").

The result will look much like this:



It is the perfect tool when you want to **test** code fast. It also has color coding and the ability to save and share code with others:

Example

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

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



HTML Basics Section 3.

we will see the **HTML Basics** by understanding all the basic stuff of HTML coding.

There are various tags that we must consider and include while starting to code in HTML. These tags help in the organization and basic formatting of elements in our script or web pages. These step-by-step procedures will guide you through the process of writing HTML.

Basic HTML Document: Below mentioned are the basic HTML tags that divide the whole document into various parts like head, body, etc.

- Every HTML document begins with a HTML document tag. Although this is not mandatory, it is a good convention to start the document with this below-mentioned tag. Please refer to the **HTML Doctypes** article for more information related to Doctypes.



<!DOCTYPE html>

HTML

 <html>

 <head>
 <!-- Information about the page -->
 <!--This is the comment tag-->

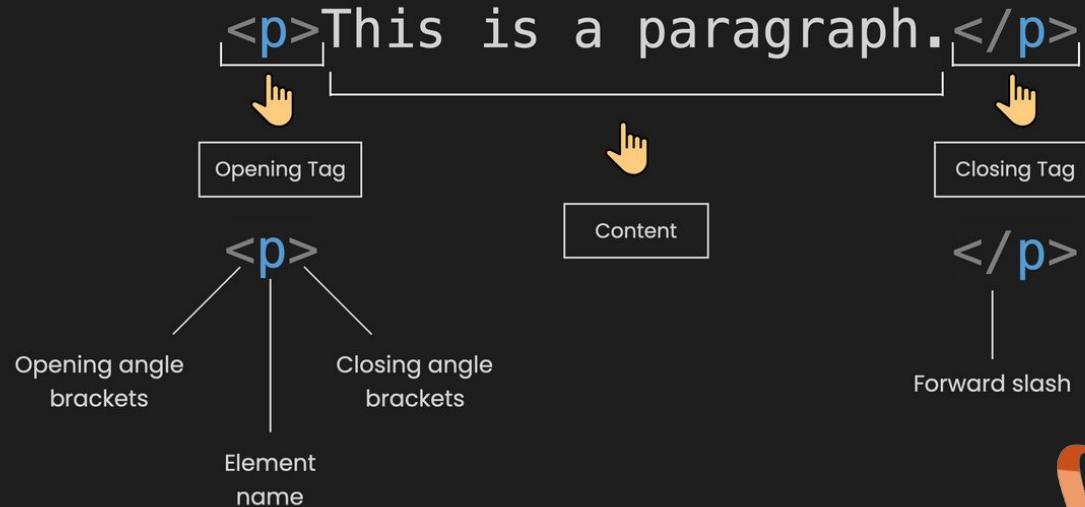
 <title>Skyline ict consult ltd</title>
 </head>

 <body>
 <!--Contents of the webpage-->
 </body>

</html>



HTML Element



The <!DOCTYPE> Declaration

The `<!DOCTYPE>` declaration represents the document type, and helps browsers to display web pages correctly.

It must only appear once, at the top of the page (before any HTML tags).

The `<!DOCTYPE>` declaration is not case sensitive.

The `<!DOCTYPE>` declaration for HTML5 is:

```
<!DOCTYPE html>
```



HTML Headings

HTML headings are defined with the `<h1>` to `<h6>` tags.

`<h1>` defines the most important heading. `<h6>` defines the least important heading:

Example

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
```



HTML Paragraphs

HTML paragraphs are defined with the `<p>` tag:

Example

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

HTML Links

HTML links are defined with the `<a>` tag:

Example

```
<a href="https://www.skylineict.com">This is a link</a>
```

The link's destination is specified in the `href` attribute.

Attributes are used to provide additional information about HTML elements.

You will learn more about attributes in a later chapter.



HTML Images

HTML images are defined with the `` tag.

The source file (`src`), alternative text (`alt`), `width`, and `height` are provided as attributes:

Example

```

```

How to View HTML Source

Have you ever seen a Web page and wondered "Hey! How did they do that?"

View HTML Source Code:

Right-click in an HTML page and select "View Page Source" (in Chrome) or "View Source" (in Edge), or similar in other browsers. This will open a window containing the HTML source code of the page.

Inspect an HTML Element:

Right-click on an element (or a blank area), and choose "Inspect" or "Inspect Element" to see what elements are made up of (you will see both the HTML and the CSS). You can also edit the HTML or CSS on-the-fly in the Elements or Styles panel that opens.

HTML Elements 4

An HTML element is defined by a starting tag. If the element contains other content, it ends with a closing tag, where the element name is preceded by a forward slash as shown below with few tags –

So here `<p>....</p>` is an HTML element, `<h1>...</h1>` is another HTML element. There are some HTML elements which don't need to be closed, such as `<img.../>`, `<hr />` and `
` elements. These are known as void elements.

HTML documents consists of a tree of these elements and they specify how HTML documents should be built, and what kind of content should be placed in what part of an HTML document.



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

HTML Elements

The **HTML element** is everything from the start tag to the end tag:

```
<tagname>Content goes here...</tagname>
```

Examples of some HTML elements:

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

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



Start tag	Element content	End tag
<h1>	My First Heading	</h1>
<p>	My first paragraph.	</p>
 	<i>none</i>	<i>none</i>

Note: Some HTML elements have no content (like the
 element). These elements are called empty elements. Empty elements do not have an end tag!

Nested HTML Elements

HTML elements can be nested (this means that elements can contain other elements).

All HTML documents consist of nested HTML elements.

The following example contains four HTML elements (`<html>`, `<body>`, `<h1>` and `<p>`):

Example

```
<!DOCTYPE html>
<html>
<body>

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

</body>
</html>
```

Example Explained

The `<html>` element is the root element and it defines the whole HTML document.

It has a start tag `<html>` and an end tag `</html>`.

Then, inside the `<html>` element there is a `<body>` element:



```
<body>

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

</body>
```

The `<body>` element defines the document's body.

It has a start tag `<body>` and an end tag `</body>`.

Then, inside the `<body>` element there are two other elements: `<h1>` and `<p>`:

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

The `<h1>` element defines a heading.

It has a start tag `<h1>` and an end tag `</h1>`:



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

The `<p>` element defines a paragraph.

It has a start tag `<p>` and an end tag `</p>`:

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

Never Skip the End Tag

Some HTML elements will display correctly, even if you forget the end tag:

Example

```
<html>
<body>

<p>This is a paragraph
<p>This is a paragraph

</body>
</html>
```

However, never rely on this! Unexpected results and errors may occur if you forget the end tag!

Empty HTML Elements

HTML elements with no content are called empty elements.

The `
` tag defines a line break, and is an empty element without a closing tag:

Example

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

HTML is Not Case Sensitive

HTML tags are not case sensitive: `<P>` means the same as `<p>`.

The HTML standard does not require lowercase tags, but W3C **recommends** lowercase in HTML, and **demands** lowercase for stricter document types like XHTML.

At W3Schools we always use lowercase tag names.



HTML Tag Reference

W3Schools' tag reference contains additional information about these tags and their attributes.

Tag	Description
<u><html></u>	Defines the root of an HTML document
<u><body></u>	Defines the document's body
<u><h1> to <h6></u>	Defines HTML headings

For a complete list of all available HTML tags, visit our [HTML Tag Reference](#).



HTML - Attributes Section 5.

We have seen few HTML tags and their usage like heading tags `<h1>`, `<h2>`, paragraph tag `<p>` and other tags. We used them so far in their simplest form, but most of the HTML tags can also have attributes, which are extra bits of information.

An attribute is used to define the characteristics of an HTML element and is placed inside the element's opening tag. All attributes are made up of two parts – a name and a value

The name is the property you want to set. For example, the paragraph `<p>` element in the example carries an attribute whose name is align, which you can use to indicate the alignment of paragraph on the page.

The value is what you want the value of the property to be set and always put within quotations. The below example shows three possible values of align attribute: left, center and right.

The src Attribute

The `` tag is used to embed an image in an HTML page. The `src` attribute specifies the path to the image to be displayed:

Example

```

```

There are two ways to specify the URL in the `src` attribute:

1. Absolute URL - Links to an external image that is hosted on another website. Example:

```
src="https://www.skylineict.com/images/img_girl.jpg".
```

Notes: External images might be under copyright. If you do not get permission to use it, you may be in violation of copyright laws. In addition, you cannot control external images; it can suddenly be removed or changed.

2. Relative URL - Links to an image that is hosted within the website. Here, the URL does not include the domain name. If the URL begins without a slash, it will be relative to the current page. Example: `src="img_girl.jpg"`. If the URL begins with a slash, it will be relative to the domain. Example: `src="/images/img_girl.jpg"`.

Tip: It is almost always best to use relative URLs. They will not break if you change domain.



The width and height Attributes

The `` tag should also contain the `width` and `height` attributes, which specify the width and height of the image (in pixels):

Example

```

```

The alt Attribute

The required `alt` attribute for the `` tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to a slow connection, or an error in the `src` attribute, or if the user uses a screen reader.

Example

```

```



Example

See what happens if we try to display an image that does not exist:

```

```



The style Attribute

The `style` attribute is used to add styles to an element, such as color, font, size, and more.

Example

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

The lang Attribute

You should always include the `lang` attribute inside the `<html>` tag, to declare the language of the Web page. This is meant to assist search engines and browsers.

The following example specifies English as the language:

```
<!DOCTYPE html>
<html lang="en">
<body>
...
</body>
</html>
```



Country codes can also be added to the language code in the `lang` attribute. So, the first two characters define the language of the HTML page, and the last two characters define the country.

The following example specifies English as the language and United States as the country:

```
<!DOCTYPE html>
<html lang="en-US">
<body>
...
</body>
</html>
```



You can see all the language codes in our [HTML Language Code Reference](#).

The title Attribute

The `title` attribute defines some extra information about an element.

The value of the title attribute will be displayed as a tooltip when you mouse over the element:

Example

```
<p title="I'm a tooltip">This is a paragraph.</p>
```

We Suggest: Always Use Lowercase Attributes

The HTML standard does not require lowercase attribute names.

The title attribute (and all other attributes) can be written with uppercase or lowercase like **title** or **TITLE**.

However, skyline ict **recommends** lowercase attributes in HTML, and **demands** lowercase attributes for stricter document types like XHTML.

We Suggest: Always Quote Attribute Values

The HTML standard does not require quotes around attribute values.

However, skyline ict **recommends** quotes in HTML, and **demands** quotes for stricter document types like XHTML.

Good:

```
<a href="https://www.skylineict.com/html/">Visit our HTML tutorial</a>
```

Bad:

```
<a href=https://www.skylineict.com/html/>Visit our HTML tutorial</a>
```

Sometimes you have to use quotes. This example will not display the title attribute correctly, because it contains a space:

Example

```
<p title=About Skylineict>
```

Single or Double Quotes?

Double quotes around attribute values are the most common in HTML, but single quotes can also be used.

In some situations, when the attribute value itself contains double quotes, it is necessary to use single quotes:

```
<p title='John "ShotGun" Nelson'>
```

Or vice versa:

```
<p title="John 'ShotGun' Nelson">
```

Chapter Summary

- All HTML elements can have **attributes**
 - The `href` attribute of `<a>` specifies the URL of the page the link goes to
 - The `src` attribute of `` specifies the path to the image to be displayed
 - The `width` and `height` attributes of `` provide size information for images
 - The `alt` attribute of `` provides an alternate text for an image
 - The `style` attribute is used to add styles to an element, such as color, font, size, and more
 - The `lang` attribute of the `<html>` tag declares the language of the Web page
 - The `title` attribute defines some extra information about an element
-

HTML - Headings Section 6.

We will know **HTML Headings**, & their implementation through examples.

An HTML heading tag is used to define the headings of a page. There are six levels of headings defined by HTML. These 6 heading elements are **h1, h2, h3, h4, h5, and h6**; with **h1** being the highest level and **h6** being the least.

- **<h1>** is used for the main heading. (Biggest in size)
- **<h2>** is used for subheadings, if there are further sections under the subheadings then the **<h3>** elements are used.
- **<h6>** for the small heading (smallest one).

Browsers display the contents of headings in different sizes. The exact size at which each browser shows the heading can vary slightly. Users can also adjust the size of the text in their browser.

HTML headings are titles or subtitles that you want to display on a webpage.

Example

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6



HTML Headings

HTML headings are defined with the `<h1>` to `<h6>` tags.

`<h1>` defines the most important heading. `<h6>` defines the least important heading.

Example

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

Headings Are Important

Search engines use the headings to index the structure and content of your web pages.

Users often skim a page by its headings. It is important to use headings to show the document structure.

`<h1>` headings should be used for main headings, followed by `<h2>` headings, then the less important `<h3>`, and so on.

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

Bigger Headings

Each HTML heading has a default size. However, you can specify the size for any heading with the `style` attribute, using the CSS `font-size` property:

Example

```
<h1 style="font-size:60px;">Heading 1</h1>
```

HTML Paragraphs Section 7.

HTML paragraph tag is used to define a paragraph in a webpage.

Let's take a simple example to see how it work. It is a notable point that a browser itself add an empty line before and after a paragraph.

An HTML **< p >** tag indicates starting of new paragraph.

its basic implementation through the examples. **The < p > tag in HTML** defines a paragraph.

These have both opening and closing tags.

So anything mentioned within **< p > and < /p >** is treated as a paragraph. Most browsers read a line as a paragraph even if we don't use the closing tag i.e, `</p>`, but this may raise unexpected results. So, it is both a good convention, and we **must** use the closing tag.



A paragraph always starts on a new line, and is usually a block of text.

HTML Paragraphs

The HTML `<p>` element defines a paragraph.

A paragraph always starts on a new line, and browsers automatically add some white space (a margin) before and after a paragraph.

Example

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

HTML Display

You cannot be sure how HTML will be displayed.

Large or small screens, and resized windows will create different results.

With HTML, you cannot change the display by adding extra spaces or extra lines in your HTML code.

The browser will automatically remove any extra spaces and lines when the page is displayed:

Example

```
<p>
This paragraph
contains a lot of lines
in the source code,
but the browser
ignores it.
</p>
```

```
<p>
This paragraph
contains      a lot of spaces
in the source      code.
```

HTML Horizontal Rules

The `<hr>` tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.

The `<hr>` element is used to separate content (or define a change) in an HTML page:

Example

```
<h1>This is heading 1</h1>
<p>This is some text.</p>
<hr>
<h2>This is heading 2</h2>
<p>This is some other text.</p>
<hr>
```

The `<hr>` tag is an empty tag, which means that it has no end tag.

HTML Line Breaks

The HTML `
` element defines a line break.

Use `
` if you want a line break (a new line) without starting a new paragraph:

Example

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

The `
` tag is an empty tag, which means that it has no end tag.



The Poem Problem

This poem will display on a single line:

Example

```
<p>
    My Bonnie lies over the ocean.
    My Bonnie lies over the sea.
    My Bonnie lies over the ocean.
    Oh, bring back my Bonnie to me.
</p>
```

Solution - The HTML <pre> Element

The HTML `<pre>` element defines preformatted text.

The text inside a `<pre>` element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks:

Example

```
<pre>
    My Bonnie lies over the ocean.

    My Bonnie lies over the sea.

    My Bonnie lies over the ocean.

    Oh, bring back my Bonnie to me.
</pre>
```

HTML Styles Section 8.

HTML Style is used to change or add the style on existing HTML elements. There is a default style for every HTML element e.g. **background color is white, text color is black etc.**

The style attribute can be used with any HTML tag. To apply style on HTML tag, you should have the basic knowledge of css properties e.g. color, background-color, text-align, font-family, font-size etc.

The syntax of style attribute is given below:

```
<h3 style="color:green">This is Green Color</h3>
```

```
<h3 style="color:red">This is Red Color</h3>
```

The HTML `style` attribute is used to add styles to an element, such as color, font, size, and more.

Example

I am Red

I am Blue

I am Big



The HTML Style Attribute

Setting the style of an HTML element, can be done with the `style` attribute.

The HTML `style` attribute has the following syntax:

```
<tagname style="property:value;">
```

The **property** is a CSS property. The **value** is a CSS value.

You will learn more about CSS later in this tutorial.

Background Color

The CSS `background-color` property defines the background color for an HTML element.

The CSS `background-color` property defines the background color for an HTML element.

Example

Set the background color for a page to powderblue:

```
<body style="background-color:powderblue;">  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>
```



Example

Set background color for two different elements:

```
<body>  
  
<h1 style="background-color:powderblue;">This is a heading</h1>  
<p style="background-color:tomato;">This is a paragraph.</p>  
  
</body>
```



Text Color

The CSS `color` property defines the text color for an HTML element:

Example

```
<h1 style="color:blue;">This is a heading</h1>  
<p style="color:red;">This is a paragraph.</p>
```

Fonts

The CSS `font-family` property defines the font to be used for an HTML element:

Example

```
<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>
```

Text Size

The CSS `font-size` property defines the text size for an HTML element:

Example

```
<h1 style="font-size:300%;">This is a heading</h1>
<p style="font-size:160%;">This is a paragraph.</p>
```

Text Alignment

The CSS `text-align` property defines the horizontal text alignment for an HTML element:

Example

```
<h1 style="text-align:center;">Centered Heading</h1>
<p style="text-align:center;">Centered paragraph.</p>
```

Chapter Summary

- Use the `style` attribute for styling HTML elements
- Use `background-color` for background color
- Use `color` for text colors
- Use `font-family` for text fonts
- Use `font-size` for text sizes
- Use `text-align` for text alignment

HTML - formatting Section 9.

HTML Formatting is a process of formatting text for better look and feel.

HTML provides us ability to format text without using CSS. There are many formatting tags in HTML. These tags are used to make **text bold, italicized, or underlined**.

There are almost 14 options available that how text appears in HTML .

In HTML the formatting tags are divided into two categories:

If you use a word processor, you must be familiar with the ability to make text bold, italicized, or underlined

- **Physical tag: These tags are used to provide the visual appearance to the text.**
- **Logical tag: These tags are used to add some logical or semantic value to the text.**

HTML Formatting Elements

Formatting elements were designed to display special types of text:

- `` - Bold text
- `` - Important text
- `<i>` - Italic text
- `` - Emphasized text
- `<mark>` - Marked text
- `<small>` - Smaller text
- `` - Deleted text
- `<ins>` - Inserted text
- `<sub>` - Subscript text
- `<sup>` - Superscript text

HTML `` and `` Elements

The HTML `` element defines bold text, without any extra importance.

Example

```
<b>This text is bold</b>
```

The HTML `` element defines text with strong importance. The content inside is typically displayed in bold.

Example

```
<strong>This text is important!</strong>
```



HTML <i> and Elements

The HTML `<i>` element defines a part of text in an alternate voice or mood. The content inside is typically displayed in italic.

Tip: The `<i>` tag is often used to indicate a technical term, a phrase from another language, a thought, a ship name, etc.

Example

```
| <i>This text is italic</i>
```



The HTML `` element defines emphasized text. The content inside is typically displayed in italic.

Tip: A screen reader will pronounce the words in `` with an emphasis, using verbal stress.

Example

```
| <em>This text is emphasized</em>
```

HTML <small> Element

The HTML `<small>` element defines smaller text:

Example

```
<small>This is some smaller text.</small>
```

HTML <mark> Element

The HTML `<mark>` element defines text that should be marked or highlighted:

Example

```
<p>Do not forget to buy <mark>milk</mark> today.</p>
```

HTML Element

The HTML `` element defines text that has been deleted from a document. Browsers will usually strike a line through deleted text:

Example

```
<p>My favorite color is <del>blue</del> red.</p>
```

HTML <ins> Element



The HTML `<ins>` element defines a text that has been inserted into a document. Browsers will usually underline inserted text:

Example

```
<p>My favorite color is <del>blue</del> <ins>red</ins>.</p>
```

HTML <sub> Element

The HTML `<sub>` element defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H₂O:

Example

```
<p>This is <sub>subscripted</sub> text.</p>
```



HTML <sup> Element

The HTML `<sup>` element defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, like WWW^[1]:

Example

```
<p>This is <sup>superscripted</sup> text.</p>
```

HTML -Quotation Section 10.

In this chapter we will go through the `<blockquote>`, `<q>`, `<abbr>`, `<address>`, `<cite>`, and `<bdo>` HTML elements.

In this chapter we will go through the `<blockquote>`, `<q>`, `<abbr>`, `<address>`, `<cite>`, and `<bdo>` HTML elements.

Example

Here is a quote from WWF's website:

For 60 years, WWF has worked to help people and nature thrive. As the world's leading conservation organization, WWF works in nearly 100 countries. At every level, we collaborate with people around the world to develop and deliver innovative solutions that protect communities, wildlife, and the places in which they live.

HTML <blockquote> for Quotations

The HTML `<blockquote>` element defines a section that is quoted from another source.

Browsers usually indent `<blockquote>` elements.

Example

```
<p>Here is a quote from WWF's website:</p>
<blockquote cite="http://www.skylineict/who/index.html">
For 60 years, WWF has worked to help people and nature thrive. As the world's leading conservation
organization, WWF works in nearly 100 countries. At every level, we collaborate with people around the
world to develop and deliver innovative solutions that protect communities, wildlife, and the places
in which they live.
</blockquote>
```



HTML <q> for Short Quotations

The HTML `<q>` tag defines a short quotation.

Browsers normally insert quotation marks around the quotation.

Example

```
<p>WWF's goal is to: <q>Build a future where people live in harmony with nature.</q></p>
```

HTML <abbr> for Abbreviations

The HTML `<abbr>` tag defines an abbreviation or an acronym, like "HTML", "CSS", "Mr.", "Dr.", "ASAP", "ATM".

Marking abbreviations can give useful information to browsers, translation systems and search-engines.

Tip: Use the global title attribute to show the description for the abbreviation/acronym when you mouse over the element.

Example

```
<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>
```

HTML <address> for Contact Information

The HTML `<address>` tag defines the contact information for the author/owner of a document or an article.

The contact information can be an email address, URL, physical address, phone number, social media handle, etc.

The text in the `<address>` element usually renders in *italic*, and browsers will always add a line break before and after the `<address>` element.

Example

```
<address>
Written by John Doe.<br>
Visit us at:<br>
Example.com<br>
Box 564, Disneyland<br>
USA
</address>
```



HTML <cite> for Work Title

The HTML `<cite>` tag defines the title of a creative work (e.g. a book, a poem, a song, a movie, a painting, a sculpture, etc.).

Note: A person's name is not the title of a work.

The text in the `<cite>` element usually renders in *italic*.

Example

```
<p><cite>The Scream</cite> by Edvard Munch. Painted in 1893.</p>
```

HTML <bdo> for Bi-Directional Override

BDO stands for Bi-Directional Override.

The HTML `<bdo>` tag is used to override the current text direction:

Example

```
<bdo dir="rtl">This text will be written from right to left</bdo>
```



HTML -Comment Tag Section 11.

HTML comments are not displayed in the browser, but they can help document your HTML source code.

HTML comments are not displayed in the browser, but they can help document your HTML source code.

HTML Comment Tag

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

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

Notice that there is an exclamation point (!) in the start tag, but not in the end tag.





Add Comments

With comments you can place notifications and reminders in your HTML code:

Example

```
<!-- This is a comment -->

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

<!-- Remember to add more information here -->
```



Hide Content

Comments can be used to hide content.

This can be helpful if you hide content temporarily:

Example

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

<!-- <p>This is another paragraph </p> -->

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

You can also hide more than one line. Everything between the `<!--` and the `-->` will be hidden from the display.

Example

Hide a section of HTML code:

```
<p>This is a paragraph.</p>
<!--
<p>Look at this cool image:</p>

-->
<p>This is a paragraph too.</p>
```



Hide Inline Content

Comments can be used to hide parts in the middle of the HTML code.

Example

Hide a part of a paragraph:

```
| <p>This <!-- great text --> is a paragraph.</p>
```

HTML -colors Section 12.

HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

Color Names

In HTML, a color can be specified by using a color name:



Tomato



Orange



DodgerBlue



MediumSeaGreen



Gray



SlateBlue



Violet



LightGray

Background Color

You can set the background color for HTML elements:

Hello World

Lore ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```



Text Color

You can set the color of text:

Hello World

Lore ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lore ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

Border Color

You can set the color of borders:

Hello World

Hello World

Hello World

Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Color Values

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values.

The following three `<div>` elements have their background color set with RGB, HEX, and HSL values:

```
rgb(255, 99, 71)
```

```
#ff6347
```

```
hsl(9, 100%, 64%)
```

The following two `<div>` elements have their background color set with RGBA and HSLA values, which add an Alpha channel to the color (here we have 50% transparency):

```
rgba(255, 99, 71, 0.5)
```

```
hsla(9, 100%, 64%, 0.5)
```

Example

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

An RGB color value represents RED, GREEN, and BLUE light sources.

An RGBA color value is an extension of RGB with an Alpha channel (opacity).

RGB Color Values

In HTML, a color can be specified as an RGB value, using this formula:

rgb(red, green, blue)

Each parameter (red, green, and blue) defines the intensity of the color with a value between 0 and 255.

This means that there are $256 \times 256 \times 256 = 16777216$ possible colors!

For example, `rgb(255, 0, 0)` is displayed as red, because red is set to its highest value (255), and the other two (green and blue) are set to 0.

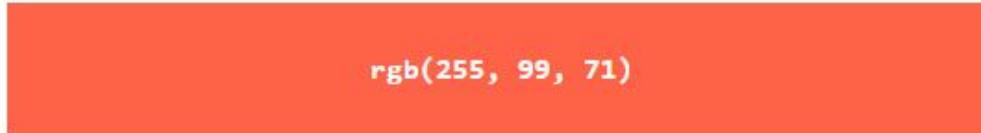
Another example, `rgb(0, 255, 0)` is displayed as green, because green is set to its highest value (255), and the other two (red and blue) are set to 0.

To display black, set all color parameters to 0, like this: `rgb(0, 0, 0)`.

To display white, set all color parameters to 255, like this: `rgb(255, 255, 255)`.

Experiment by mixing the RGB values below:





RED



255

GREEN



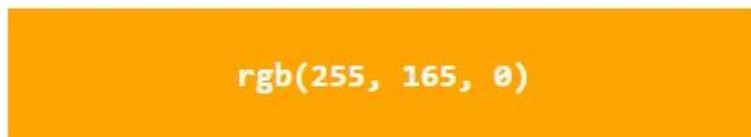
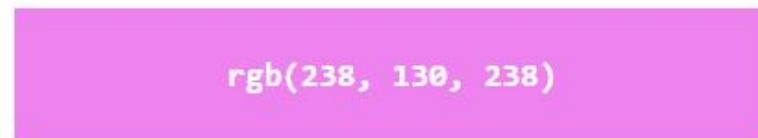
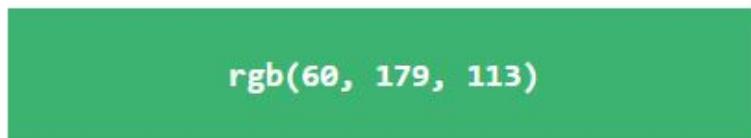
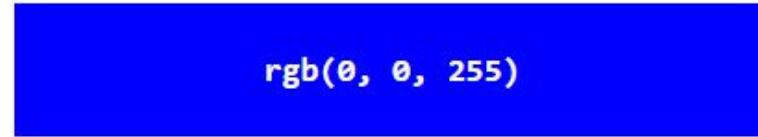
99

BLUE



71

Example



HTML - Styles - CSS Section 13.

HTML (HyperText Markup Language) and CSS (Cascading Style Sheet) are the fundamental web development languages.

HTML defines a website's content and structure, while CSS specifies the design and presentation. Together, both languages allow to create a website that is well-structured and functional.

CSS defines style declarations and applies them to HTML documents. There are three different ways to link CSS to HTML

- **Inline** – uses the **style** attribute inside an HTML element
- **Internal** – written in the **<head>** section of an HTML file
- **External** – links an HTML document to an external CSS file

CSS = Styles and Colors

M a n i p u l a t e T e x t
C o l o r s , B o x e s

What is CSS?

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

Tip: The word **cascading** means that a style applied to a parent element will also apply to all children elements within the parent. So, if you set the color of the body text to "blue", all headings, paragraphs, and other text elements within the body will also get the same color (unless you specify something else)!

Using CSS

CSS can be added to HTML documents in 3 ways:

- **Inline** - by using the `style` attribute inside HTML elements
- **Internal** - by using a `<style>` element in the `<head>` section
- **External** - by using a `<link>` element to link to an external CSS file

The most common way to add CSS, is to keep the styles in external CSS files. However, in this tutorial we will use inline and internal styles, because this is easier to demonstrate, and easier for you to try it yourself.

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the `style` attribute of an HTML element.

The following example sets the text color of the `<h1>` element to blue, and the text color of the `<p>` element to red:

Example

```
<h1 style="color:blue;">A Blue Heading</h1>  
  
<p style="color:red;">A red paragraph.</p>
```

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the `<head>` section of an HTML page, within a `<style>` element.

The following example sets the text color of ALL the `<h1>` elements (on that page) to blue, and the text color of ALL the `<p>` elements to red. In addition, the page will be displayed with a "powderblue" background color:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1  {color: blue;}
p   {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

External CSS

An external style sheet is used to define the `style` for many HTML pages.

To use an external style sheet, add a link to it in the `<head>` section of each HTML page:

Example

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is what the "styles.css" file looks like:

"styles.css":

```
body {  
    background-color: powderblue;  
}  
h1 {  
    color: blue;  
}  
p {  
    color: red;  
}
```



Tip: With an external style sheet, you can change the look of an entire web site, by changing one file!

CSS Colors, Fonts and Sizes

Here, we will demonstrate some commonly used CSS properties. You will learn more about them later.

The CSS `color` property defines the text color to be used.

The CSS `font-family` property defines the font to be used.

The CSS `font-size` property defines the text size to be used.

Example

Use of CSS color, font-family and font-size properties:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>
```

CSS Border

The CSS `border` property defines a border around an HTML element.

Tip: You can define a border for nearly all HTML elements.

Example

Use of CSS border property:

```
p {  
    border: 2px solid powderblue;  
}
```

CSS Padding

The CSS `padding` property defines a padding (space) between the text and the border.

Example

Use of CSS border and padding properties:

```
p {  
    border: 2px solid powderblue;  
    padding: 30px;  
}
```

CSS Margin

The CSS `margin` property defines a margin (space) outside the border.

Example

Use of CSS border and margin properties:

```
p {  
    border: 2px solid powderblue;  
    margin: 50px;  
}
```

Link to External CSS

External style sheets can be referenced with a full URL or with a path relative to the current web page.

Example

This example uses a full URL to link to a style sheet:

```
<link rel="stylesheet" href="https://www.skylineict.com/html/styles.css">
```

Example

This example links to a style sheet located in the html folder on the current web site:

```
<link rel="stylesheet" href="/html/styles.css">
```

Example

This example links to a style sheet located in the same folder as the current page:

```
<link rel="stylesheet" href="styles.css">
```

Chapter Summary

- Use the HTML `style` attribute for inline styling
- Use the HTML `<style>` element to define internal CSS
- Use the HTML `<link>` element to refer to an external CSS file
- Use the HTML `<head>` element to store `<style>` and `<link>` elements
- Use the CSS `color` property for text colors
- Use the CSS `font-family` property for text fonts
- Use the CSS `font-size` property for text sizes
- Use the CSS `border` property for borders
- Use the CSS `padding` property for space inside the border
- Use the CSS `margin` property for space outside the border

HTML - Links Section 14.

A link or hyperlink is a connection from one web resource to another.

Links allow users to move seamlessly from one page to another, on any server anywhere in the world.

A link has two ends, called anchors. The link starts at the source anchor and points to the destination anchor, which may be any web resource, for example, an image, an audio or video clip, a PDF file, an HTML document or an element within the document itself, and so on.

By default, links will appear as follow in most of the browsers:

- An unvisited link is underlined and blue.
- A visited link is underlined and purple.
- An active link is underlined and red.

Links are found in nearly all web pages. Links allow users to click their way from page to page.

HTML Links - Hyperlinks

HTML links are hyperlinks.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand.

Note: A link does not have to be text. A link can be an image or any other HTML element!

HTML Links - Syntax

The HTML `<a>` tag defines a hyperlink. It has the following syntax:

```
| <a href="url">Link text</a>
```



The HTML `<a>` tag defines a hyperlink. It has the following syntax:

```
<a href="url">link text</a>
```

The most important attribute of the `<a>` element is the `href` attribute, which indicates the link's destination.

The *link text* is the part that will be visible to the reader.

Clicking on the link text, will send the reader to the specified URL address.

Example

This example shows how to create a link to Skyline.com:

```
<a href="https://www.skylineict.com/">Visit skyline.com!</a>
```



HTML Links - The target Attribute

By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.

The `target` attribute specifies where to open the linked document.

The `target` attribute can have one of the following values:

- `_self` - Default. Opens the document in the same window/tab as it was clicked
- `_blank` - Opens the document in a new window or tab
- `_parent` - Opens the document in the parent frame
- `_top` - Opens the document in the full body of the window



Example

Use `target="_blank"` to open the linked document in a new browser window or tab:

```
<a href="https://www.skyline.com/" target="_blank">Visit skyline ict consult ltd!</a>
```

Absolute URLs vs. Relative URLs

Both examples above are using an **absolute URL** (a full web address) in the `href` attribute.

A local link (a link to a page within the same website) is specified with a **relative URL** (without the "https://www" part):

Example

```
<h2>Absolute URLs</h2>
<p><a href="https://www.skylineict/">skyline</a></p>
<p><a href="https://www.google.com/">Google</a></p>

<h2>Relative URLs</h2>
<p><a href="html_images.asp">HTML Images</a></p>
<p><a href="/css/default.asp">CSS Tutorial</a></p>
```

HTML Links - Use an Image as a Link

To use an image as a link, just put the `` tag inside the `<a>` tag:

Example

```
<a href="default.asp">  
    
</a>
```

Link to an Email Address

Use `mailto:` inside the `href` attribute to create a link that opens the user's email program (to let them send a new email):

Example

```
<a href="mailto:student@skylineict.com">Send email</a>
```

Button as a Link

To use an HTML button as a link, you have to add some JavaScript code.

JavaScript allows you to specify what happens at certain events, such as a click of a button:

Example

```
<button onclick="document.location='default.asp'">HTML Tutorial</button>
```

Link Titles

The `title` attribute specifies extra information about an element. The information is most often shown as a tooltip text when the mouse moves over the element.

Example

```
<a href="https://www.skylineict.com/html/" title="Skyline ict consult lt d">Visit our HTML Tutorial</a>
```

Example

Link to a page located in the html folder on the current web site:

```
<a href="/html/default.asp">HTML tutorial</a>
```

Example

Link to a page located in the same folder as the current page:

```
<a href="default.asp">HTML tutorial</a>
```

Chapter Summary

- Use the `<a>` element to define a link
 - Use the `href` attribute to define the link address
 - Use the `target` attribute to define where to open the linked document
 - Use the `` element (inside `<a>`) to use an image as a link
 - Use the `mailto:` scheme inside the `href` attribute to create a link that opens the user's email program
-

Example

Here, an unvisited link will be green with no underline. A visited link will be pink with no underline. An active link will be yellow and underlined. In addition, when mousing over a link (a:hover) it will become red and underlined:

```
<style>
a:link {
    color: green;
    background-color: transparent;
    text-decoration: none;
}

a:visited {
    color: pink;
    background-color: transparent;
    text-decoration: none;
}

a:hover {
    color: red;
    background-color: transparent;
    text-decoration: underline;
}
```





Link Buttons

A link can also be styled as a button, by using CSS:

This is a link

Example

```
<style>
a:link, a:visited {
    background-color: #f44336;
    color: white;
    padding: 15px 25px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
}
```

HTML - images Section 15.

Images enhance visual appearance of the web pages by making them more interesting and colorful.

The `` tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The syntax of the `` tag can be given with:

Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. This tutorial will take you through simple steps to use images in your web pages.

Example

```

```

Example

```

```

HTML Images Syntax

The HTML `` tag is used to embed an image in a web page.

Images are not technically inserted into a web page; images are linked to web pages. The `` tag creates a holding space for the referenced image.

The `` tag is empty, it contains attributes only, and does not have a closing tag.

The `` tag has two required attributes:

- `src` - Specifies the path to the image
- `alt` - Specifies an alternate text for the image



Syntax

```

```

The src Attribute

The required `src` attribute specifies the path (URL) to the image.

Note: When a web page loads, it is the browser, at that moment, that gets the image from a web server and inserts it into the page. Therefore, make sure that the image actually stays in the same spot in relation to the web page, otherwise your visitors will get a broken link icon. The broken link icon and the `alt` text are shown if the browser cannot find the image.

Example

```

```



The alt Attribute

The required `alt` attribute provides an alternate text for an image, if the user for some reason cannot view it (because of a slow connection, an error in the `src` attribute, or if the user uses a screen reader).

The value of the `alt` attribute should describe the image:

Example

```

```

If a browser cannot find an image, it will display the value of the `alt` attribute:

Example

```

```

Image Size - Width and Height

You can use the `style` attribute to specify the width and height of an image.

Example

```

```

Alternatively, you can use the `width` and `height` attributes:

Example

```

```

Width and Height, or Style?

The `width`, `height`, and `style` attributes are all valid in HTML.

However, we suggest using the `style` attribute. It prevents styles sheets from changing the size of images:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  width: 100%;
}
</style>
</head>
<body>





</body>
```



Images in Another Folder

If you have your images in a sub-folder, you must include the folder name in the `src` attribute:



Example

```

```

Images on Another Server/Website

Some web sites point to an image on another server.

To point to an image on another server, you must specify an absolute (full) URL in the `src` attribute:

Example

```

```

Images in Another Folder

If you have your images in a sub-folder, you must include the folder name in the `src` attribute:

Example

```

```



Images on Another Server/Website

Some web sites point to an image on another server.

To point to an image on another server, you must specify an absolute (full) URL in the `src` attribute:

Example

```

```

Animated Images

HTML allows animated GIFs:

Example

```

```



Image as a Link

To use an image as a link, put the `` tag inside the `<a>` tag:

Example

```
<a href="default.asp">
  
</a>
```

Image Floating

Use the CSS `float` property to let the image float to the right or to the left of a text:

Example

```
<p>  
The image will float to the right of the text.</p>
```

```
<p>  
The image will float to the left of the text.</p>
```



Common Image Formats

Here are the most common image file types, which are supported in all browsers (Chrome, Edge, Firefox, Safari, Opera):

Abbreviation	File Format	File Extension
APNG	Animated Portable Network Graphics	.apng
GIF	Graphics Interchange Format	.gif
ICO	Microsoft Icon	.ico, .cur
JPEG	Joint Photographic Expert Group image	.jpg, .jpeg, .jfif, .pjpeg, .ppj
PNG	Portable Network Graphics	.png
SVG	Scalable Vector Graphics	.svg

Chapter Summary

- Use the HTML `` element to define an image
- Use the HTML `src` attribute to define the URL of the image
- Use the HTML `alt` attribute to define an alternate text for an image, if it cannot be displayed
- Use the HTML `width` and `height` attributes or the CSS `width` and `height` properties to define the size of the image
- Use the CSS `float` property to let the image float to the left or to the right

A background image can be specified for almost any **HTML element**.

Background Image on a HTML element

To add a background image on an HTML element, use the **HTML `style`** attribute and the **CSS `background-image`** property:

Example

Add a background image on a HTML element:

```
<p style="background-image: url('img_girl.jpg');">
```

Example

Specify the background image in the `<style>` element:

```
<style>
p {
  background-image: url('img_girl.jpg');
}
</style>
```

Background Image on a Page

If you want the entire page to have a background image, you must specify the background image on the `<body>` element:

Example

Add a background image for the entire page:

```
<style>
body {
  background-image: url('img_girl.jpg');
}
</style>
```



Background Repeat

If the background image is smaller than the element, the image will repeat itself, horizontally and vertically, until it reaches the end of the element:



Example

```
<style>
body {
  background-image: url('example_img_girl.jpg');
}
</style>
```



To avoid the background image from repeating itself, set the `background-repeat` property to `no-repeat`.

Example

```
<style>
body {
    background-image: url('example_img_girl.jpg');
    background-repeat: no-repeat;
}
</style>
```

Background Cover

If you want the background image to cover the entire element, you can set the `background-size` property to `cover`.

Also, to make sure the entire element is always covered, set the `background-attachment` property to `fixed`:

This way, the background image will cover the entire element, with no stretching (the image will keep its original proportions):

Example

```
<style>
body {
    background-image: url('img_girl.jpg');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: cover;
}
</style>
```



Background Stretch

If you want the background image to stretch to fit the entire element, you can set the `background-size` property to `100% 100%`:



Try resizing the browser window, and you will see that the image will stretch, but always cover the entire element.

Example

```
<style>
body {
    background-image: url('img_girl1.jpg');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: 100% 100%;
}
</style>
```



The HTML `<picture>` element allows you to display different pictures for different devices or screen sizes.



The HTML <picture> Element

The HTML `<picture>` element gives web developers more flexibility in specifying image resources.

The `<picture>` element contains one or more `<source>` elements, each referring to different images through the `srcset` attribute. This way the browser can choose the image that best fits the current view and/or device.

Each `<source>` element has a `media` attribute that defines when the image is the most suitable.

Example

Show different images for different screen sizes:

```
<picture>
  <source media="(min-width: 650px)" srcset="img_food.jpg">
  <source media="(min-width: 465px)" srcset="img_car.jpg">
  
</picture>
```



When to use the Picture Element

There are two main purposes for the `<picture>` element:

1. Bandwidth

If you have a small screen or device, it is not necessary to load a large image file. The browser will use the first `<source>` element with matching attribute values, and ignore any of the following elements.

2. Format Support

Some browsers or devices may not support all image formats. By using the `<picture>` element, you can add images of all formats, and the browser will use the first format it recognizes, and ignore any of the following elements.

Example

The browser will use the first image format it recognizes:

```
<picture>
  <source srcset="img_avatar.png">
  <source srcset="img_girl.jpg">
  
</picture>
```



HTML - Favicon Section 16.



simply put, a favicon is your website logo that appears next to the **meta title** on your browser tab. In other words, instead of showing a blank document icon on the browser, your website will display your official website icon.

A favicon is a small image displayed next to the page title in the browser tab.

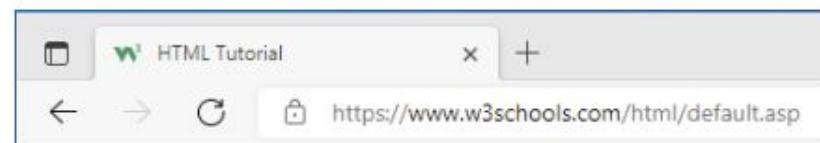


How To Add a Favicon in HTML

You can use any image you like as your favicon. You can also create your own favicon on sites like <https://www.favicon.cc>.

Tip: A favicon is a small image, so it should be a simple image with high contrast.

A favicon image is displayed to the left of the page title in the browser tab, like this:



To add a favicon to your website, either save your favicon image to the root directory of your webserver, or create a folder in the root directory called images, and save your favicon image in this folder. A common name for a favicon image is "favicon.ico".

Next, add a `<link>` element to your "index.html" file, after the `<title>` element, like this:

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>My Page Title</title>
    <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
<body>

    <h1>This is a Heading</h1>
    <p>This is a paragraph.</p>

</body>
</html>
```

Now, save the "index.html" file and reload it in your browser. Your browser tab should now display your favicon image to the left of the page title.

HTML Tables Section 17.

In this section we will know the **HTML Table**, various ways to implement it, & will also understand its usage through the examples.

HTML Table is an arrangement of data in rows and columns, or possibly in a more complex structure.

Tables are widely used in communication, research, and data analysis. Tables are useful for various tasks such as presenting text information and numerical data. It can be used to compare two or more items in the tabular form layout. Tables are used to create databases.

Defining Tables in HTML: An HTML table is defined with the “table” tag. Each table row is defined with the “tr” tag. A table header is defined with the “th” tag. By default, table headings are bold and centered. A table data/cell is defined with the “td” tag.

HTML tables allow web developers to arrange data into rows and columns.

Example

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Define an HTML Table

A table in HTML consists of table cells inside rows and columns.

Example

A simple HTML table:

```
<table>
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
</table>
```



Table Cells

Each table cell is defined by a `<td>` and a `</td>` tag.

`td` stands for table data.

Everything between `<td>` and `</td>` are the content of the table cell.

Example

```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
</table>
```

Table Rows

Each table row starts with a `<tr>` and ends with a `</tr>` tag.

`tr` stands for table row.

Example

```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
  <tr>
    <td>16</td>
    <td>14</td>
    <td>10</td>
  </tr>
</table>
```



Table Headers

Sometimes you want your cells to be table header cells. In those cases use the `<th>` tag instead of the `<td>` tag:

`th` stands for table header.

Example

Let the first row be table header cells:

```
<table>
  <tr>
    <th>Person 1</th>
    <th>Person 2</th>
    <th>Person 3</th>
  </tr>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
  <tr>
    <td>16</td>
    <td>14</td>
    <td>10</td>
  </tr>
</table>
```



HTML tables can have borders of different styles and shapes.

How To Add a Border

When you add a border to a table, you also add borders around each table cell:

To add a border, use the CSS `border` property on `table`, `th`, and `td` elements:

Example

```
table, th, td {  
    border: 1px solid black;  
}
```



Collapsed Table Borders

To avoid having double borders like in the example above, set the CSS `border-collapse` property to `collapse`.

This will make the borders collapse into a single border:

Example

```
table, th, td {  
    border: 1px solid black;  
    border-collapse: collapse;  
}
```

Style Table Borders

If you set a background color of each cell, and give the border a white color (the same as the document background), you get the impression of an invisible border:

Example

```
table, th, td {  
    border: 1px solid white;  
    border-collapse: collapse;  
}  
th, td {  
    background-color: #96D4D4;  
}
```

HTML tables can have different sizes for each column, row or the entire table.

Use the `style` attribute with the `width` or `height` properties to specify the size of a table, row or column.

HTML Table Width

To set the width of a table, add the `style` attribute to the `<table>` element:

HTML tables can have headers for each column or row, or for many columns/rows.

EMIL	TOBIAS	LINUS

8:00		
9:00		
10:00		
11:00		
12:00		
13:00		

	MON	TUE	WED	THU	FRI
8:00					
9:00					
10:00					
11:00					
12:00					

DECEMBER					

HTML Table Headers

Table headers are defined with `th` elements. Each `th` element represents a table cell.

Example

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```



Vertical Table Headers

To use the first column as table headers, define the first cell in each row as a `<th>` element:

Example

```
<table>
  <tr>
    <th>Firstname</th>
    <td>Jill</td>
    <td>Eve</td>
  </tr>
  <tr>
    <th>Lastname</th>
    <td>Smith</td>
    <td>Jackson</td>
  </tr>
  <tr>
    <th>Age</th>
    <td>94</td>
    <td>50</td>
  </tr>
</table>
```



Align Table Headers

By default, table headers are bold and centered:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94



To left-align the table headers, use the CSS `text-align` property:

Example

```
th {  
    text-align: left;  
}
```

Header for Multiple Columns

You can have a header that spans over two or more columns.

Name		Age
Jill	Smith	50
Eve	Jackson	94

To do this, use the `colspan` attribute on the `<th>` element:



Example

```
<table>
  <tr>
    <th colspan="2">Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```



Table Caption

You can add a caption that serves as a heading for the entire table.

Monthly savings	
Month	Savings
January	\$100
February	\$50

To add a caption to a table, use the `<caption>` tag:

Example

```
<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

HTML tables can adjust the padding inside the cells, and also the space between the cells.

With Padding		
hello	hello	hello
hello	hello	hello
hello	hello	hello

With Spacing		
hello	hello	hello
hello	hello	hello
hello	hello	hello



HTML Table - Cell Padding

Cell padding is the space between the cell edges and the cell content.

By default the padding is set to 0.

To add padding on table cells, use the CSS `padding` property:

Example

```
th, td {  
    padding: 15px;  
}
```

To add padding only above the content, use the `padding-top` property.

And the others sides with the `padding-bottom`, `padding-left`, and `padding-right` properties:

Example

```
th, td {  
    padding-top: 10px;  
    padding-bottom: 20px;  
    padding-left: 30px;  
    padding-right: 40px;  
}
```



HTML Table - Cell Spacing

Cell spacing is the space between each cell.

By default the space is set to 2 pixels.

To change the space between table cells, use the CSS `border-spacing` property on the `table` element:

Example

```
table {  
    border-spacing: 30px;  
}
```

Use CSS to make your tables look better.

HTML Table - Zebra Stripes

If you add a background color on every other table row, you will get a nice zebra stripes effect.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

To style every other table row element, use the `:nth-child(even)` selector like this:

Example

```
tr:nth-child(even) {  
    background-color: #D6EEEE;  
}
```



Hoverable Table

Use the `:hover` selector on `tr` to highlight table rows on mouse over:

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
tr:hover {background-color: #D6EEEE;}
```

Try it yourself

HTML Lists Section 18.

In this article, we will know the **HTML List**, along with understanding its types, and various ways to implement them, through the example.

A list is a record of short pieces of related information or used to display the data or any information on web pages in the ordered or unordered form.

For instance, to purchase the items, we need to prepare a list that can either be ordered or unordered list which helps us to organize the data & easy to find the item.

HTML lists allow web developers to group a set of related items in lists.

Example

An unordered HTML list:

- Item
- Item
- Item
- Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

Unordered HTML List

An unordered list starts with the `` tag. Each list item starts with the `` tag.

The list items will be marked with bullets (small black circles) by default:

Example

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Ordered HTML List

An ordered list starts with the `` tag. Each list item starts with the `` tag.

The list items will be marked with numbers by default:

Example

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

HTML Description Lists

HTML also supports description lists.

A description list is a list of terms, with a description of each term.

The `<dl>` tag defines the description list, the `<dt>` tag defines the term (name), and the `<dd>` tag describes each term:

Example

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

HTML List Tags

Tag	Description
<u></u>	Defines an unordered list
<u></u>	Defines an ordered list
<u></u>	Defines a list item
<u><dl></u>	Defines a description list
<u><dt></u>	Defines a term in a description list
<u><dd></u>	Describes the term in a description list

HTML Block and Inline Elements Section 18.

The inline and block elements of HTML are one of the important areas where web developers often get confused because they were unable to know which are inline and block elements which may cause clumsiness in a web page in case he assumes some element to be a block but it is an inline element which causes next element comes next to it.

So let us see the differences between the inline and block elements in HTML and the different frequently used inline and block HTML elements.

Block elements: They consume the entire width available irrespective of their sufficiency. They always start in a new line and have top and bottom margins. It does not contain any other elements next to it.

Every HTML element has a default display value, depending on what type of element it is.

There are two display values: block and inline.

Block-level Elements

A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.

A block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Two commonly used block elements are: `<p>` and `<div>`.

The `<p>` element defines a paragraph in an HTML document.

The `<div>` element defines a division or a section in an HTML document.

The `<p>` element is a block-level element.

The `<div>` element is a block-level element.

Example

```
<p>Hello World</p>
<div>Hello World</div>
```

Here are the block-level elements in HTML:

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

Inline Elements

An inline element does not start on a new line.

An inline element only takes up as much width as necessary.

This is a `` element inside a paragraph.

Example

```
<span>Hello World</span>
```

Here are the inline elements in HTML:

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

The <div> Element

The `<div>` element is often used as a container for other HTML elements.

The `<div>` element has no required attributes, but `style`, `class` and `id` are common.

When used together with CSS, the `<div>` element can be used to style blocks of content:

Example

```
<div style="background-color:black;color:white;padding:20px;">
  <h2>London</h2>
  <p>London is the capital city of England. It is the most populous city in the United Kingdom, with a
metropolitan area of over 13 million inhabitants.</p>
</div>
```



The Element

The `` element is an inline container used to mark up a part of a text, or a part of a document.

The `` element has no required attributes, but `style`, `class` and `id` are common.

When used together with CSS, the `` element can be used to style parts of the text:

Example

```
<p>My mother has <span style="color:blue;font-weight:bold;">blue</span> eyes and my father has <span style="color:darkolivegreen;font-weight:bold;">dark green</span> eyes.</p>
```

Chapter Summary

- There are two display values: `block` and `inline`
- A `block-level` element always `starts` on a new `line` and `takes up` the full `width` available
- An `inline` element does not `start` on a new `line` and it only `takes up` as much `width` as necessary
- The `<div>` element is a `block-level` and is often used as a container for other `HTML` elements
- The `` element is an `inline` container used to mark up a part of a text, or a part of a document

HTML Tags

Tag	Description
<code><div></code>	Defines a section in a document (block-level)
<code></code>	Defines a section in a document (inline)

HTML Classes section 20.

The HTML class attribute is used to specify a single or multiple class names for an HTML element.

The class name can be used by **CSS and JavaScript** to do some tasks for HTML elements. You can use this class in CSS with a specific class, write a period (.) character, followed by the name of the class for selecting elements.

A class attribute can be defined within `<style>` tag or in separate file using the (.) character.

In an HTML document, we can use the same class attribute name with different elements.

the class is an attribute which specifies one or more class names for an HTML element.

The class attribute can be used on any HTML element.

The class name can be used by CSS and JavaScript to perform certain tasks for elements with the specified class name.

The HTML `class` attribute is used to specify a class for an HTML element.

Multiple HTML elements can share the same class.

Using The `class` Attribute

The `class` attribute is often used to point to a class name in a style sheet. It can also be used by a JavaScript to access and manipulate elements with the specific class name.

In the following example we have three `<div>` elements with a `class` attribute with the value of "city". All of the three `<div>` elements will be styled equally according to the `.city` style definition in the head section:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.city {
    background-color: tomato;
    color: white;
    border: 2px solid black;
    margin: 20px;
    padding: 20px;
}
</style>
</head>
<body>

<div class="city">
    <h2>London</h2>
    <p>London is the capital of England.</p>
</div>

<div class="city">
    <h2>Paris</h2>
    <p>Paris is the capital of France.</p>
</div>
```

Activate W
Go to Settings

```
<div class="city">  
  <h2>Tokyo</h2>  
  <p>Tokyo is the capital of Japan.</p>  
</div>  
  
</body>  
</html>
```

In the following example we have two `` elements with a `class` attribute with the value of "note". Both `` elements will be styled equally according to the `.note` style definition in the head section:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.note {
  font-size: 120%;
  color: red;
}
</style>
</head>
<body>

<h1>My <span class="note">Important</span> Heading</h1>
<p>This is some <span class="note">important</span> text.</p>

</body>
</html>
```

The Syntax For Class

To create a class; write a period (.) character, followed by a class name. Then, define the CSS properties within curly braces {}:

Example

Create a class named "city":

```
<!DOCTYPE html>
<html>
<head>
<style>
.city {
    background-color: tomato;
    color: white;
    padding: 10px;
}
</style>
</head>
<body>
```



```
<h2 class="city">London</h2>
<p>London is the capital of England.</p>

<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>

</body>
</html>
```

Multiple Classes

HTML elements can belong to more than one class.

To define multiple classes, separate the class names with a space, e.g. `<div class="city main">`. The element will be styled according to all the classes specified.

In the following example, the first `<h2>` element belongs to both the `city` class and also to the `main` class, and will get the CSS styles from both of the classes:

Example

```
<h2 class="city main">London</h2>
<h2 class="city">Paris</h2>
<h2 class="city">Tokyo</h2>
```

Different Elements Can Share Same Class

Different HTML elements can point to the same class name.

In the following example, both `<h2>` and `<p>` points to the "city" class and will share the same style:

Example

```
<h2 class="city">Paris</h2>
<p class="city">Paris is the capital of France</p>
```

Use of The class Attribute in JavaScript

The class name can also be used by JavaScript to perform certain tasks for specific elements.

JavaScript can access elements with a specific class name with the `getElementsByName()` method:

Example

Click on a button to hide all elements with the class name "city":

```
<script>
function myFunction() {
  var x = document.getElementsByClassName("city");
  for (var i = 0; i < x.length; i++) {
    x[i].style.display = "none";
  }
}
</script>
```

Chapter Summary

- The HTML `class` attribute specifies one or more class names for an element
- Classes are used by CSS and JavaScript to select and access specific elements
- The `class` attribute can be used on any HTML element
- The class name is case sensitive
- Different HTML elements can point to the same class name
- JavaScript can access elements with a specific class name with the `getElementsByClassName()` method

HTML ID Section 21.

In this section we will know how to identify the specific HTML element by its *id* using **HTML id Attribute**,

along with understanding its implementation through the examples.

The **id attribute** is a unique identifier that is used to specify the document. It is used by CSS and JavaScript to perform a certain task for a unique element. In CSS, the **id** attribute is used using the **#** symbol followed by id. quotes are not mandatory in tag=" " in all cases. But writing with quotes is a good practice.

The **id** attribute on an element assigns an identifier to that element.

The identifier must be unique across the page.

The **id** is a global attribute that can be applied to any HTML element.

The HTML `id` attribute is used to specify a unique id for an HTML element.

You cannot have more than one element with the same `id` in an HTML document.

Using The `id` Attribute

The `id` attribute specifies a unique id for an HTML element. The value of the `id` attribute must be unique within the HTML document.

The `id` attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

The syntax for id is: write a hash character (#), followed by an id name. Then, define the CSS properties within curly braces {}.

In the following example we have an `<h1>` element that points to the id name "myHeader". This `<h1>` element will be styled according to the `#myHeader` style definition in the head section:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
    background-color: lightblue;
    color: black;
    padding: 40px;
    text-align: center;
}
</style>
</head>
<body>

<h1 id="myHeader">My Header</h1>

</body>
</html>
```

Difference Between Class and ID

A class name can be used by multiple HTML elements, while an id name must only be used by one HTML element within the page:

Example

```
<style>
/* Style the element with the id "myHeader" */
#myHeader {
    background-color: lightblue;
    color: black;
    padding: 40px;
    text-align: center;
}

/* Style all elements with the class name "city" */
.city {
    background-color: tomato;
    color: white;
    padding: 10px;
}
</style>
```



```
<!-- An element with a unique id -->
<h1 id="myHeader">My Cities</h1>

<!-- Multiple elements with same class -->
<h2 class="city">London</h2>
<p>London is the capital of England.</p>

<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>
```

HTML Bookmarks with ID and Links

HTML bookmarks are used to allow readers to jump to specific parts of a webpage.

Bookmarks can be useful if your page is very long.

To use a bookmark, you must first create it, and then add a link to it.

Then, when the link is clicked, the page will scroll to the location with the bookmark.

Example

First, create a bookmark with the `id` attribute:

```
<h2 id="C4">Chapter 4</h2>
```

Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:

```
<a href="html_demo.html#C4">Jump to Chapter 4</a>
```

Using The id Attribute in JavaScript

The `id` attribute can also be used by JavaScript to perform some tasks for that specific element.

JavaScript can access an element with a specific id with the `getElementById()` method:

Example

Use the `id` attribute to manipulate text with JavaScript:

```
<script>
function displayResult() {
    document.getElementById("myHeader").innerHTML = "Have a nice day!";
}
</script>
```

Chapter Summary

- The `id` attribute is used to specify a unique id for an HTML element
- The value of the `id` attribute must be unique within the HTML document
- The `id` attribute is used by CSS and JavaScript to style/select a specific element
- The value of the `id` attribute is case sensitive
- The `id` attribute is also used to create HTML bookmarks
- JavaScript can access an element with a specific id with the `getElementById()` method

HTML Iframe Syntax 22.

HTML Iframe is used to display a nested webpage (a webpage within a webpage). The HTML **<iframe> tag** defines an inline frame, hence it is also called as an Inline frame.

An HTML iframe embeds another document within the current HTML document in the rectangular region.

The webpage content and iframe contents can interact with each other using JavaScript.

Iframe Syntax

An HTML iframe is defined with the **<iframe> tag**:

HTML Iframe Syntax

The HTML `<iframe>` tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

Syntax

```
| <iframe src="url" title="description"></iframe>
```



Tip: It is a good practice to always include a `title` attribute for the `<iframe>`. This is used by screen readers to read out what the content of the iframe is.

Iframe - Set Height and Width

Use the `height` and `width` attributes to specify the size of the iframe.

The height and width are specified in pixels by default:

Example

```
<iframe src="demo_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>
```

Or you can add the `style` attribute and use the CSS `height` and `width` properties:

Example

```
<iframe src="demo_iframe.htm" style="height:200px;width:300px;" title="Iframe Example"></iframe>
```



Iframe - Remove the Border

By default, an `iframe` has a border around it.

To remove the border, add the `style` attribute and use the CSS `border` property:

Example

```
<iframe src="demo_iframe.htm" style="border:none;" title="Iframe Example"></iframe>
```

With CSS, you can also change the size, style and color of the iframe's border:

Example

```
<iframe src="demo_iframe.htm" style="border:2px solid red;" title="Iframe Example"></iframe>
```

Iframe - Target for a Link

An iframe can be used as the target frame for a link.

The `target` attribute of the link must refer to the `name` attribute of the iframe:

Example

```
<iframe src="demo_iframe.htm" name="iframe_a" title="Iframe Example"></iframe>

<p><a href="https://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
```

Chapter Summary

- The HTML `<iframe>` tag specifies an inline frame
 - The `src` attribute defines the URL of the page to embed
 - Always include a `title` attribute (for screen readers)
 - The `height` and `width` attributes specifies the size of the iframe
 - Use `border:none;` to remove the border around the iframe
-

HTML File Paths 22.

A file path specifies the location of a file inside a web folder structure. It's like an address of a file which helps the web browser to access the files.

File paths are used to link external resources such as images, videos, style sheets, JavaScript, displaying other web pages etc.

To insert a file in a web page its source must be known. For example, the syntax (``) is used to insert an image file, where the path of the file is mentioned in the source (src).

File paths are of two types:

- **Absolute File Paths**
- **Relative File Paths**

A file path describes the location of a file in a web site's folder structure.

File Path Examples



Path	Description
	The "picture.jpg" file is located in the same folder as the current page
	The "picture.jpg" file is located in the images folder in the current folder
	The "picture.jpg" file is located in the images folder at the root of the current web
	The "picture.jpg" file is located in the folder one level up from the current folder

HTML File Paths

A file path describes the location of a file in a web site's folder structure.

File paths are used when linking to external files, like:

- Web pages
- Images
- Style sheets
- JavaScripts



Absolute File Paths

An absolute file path is the full URL to a file:

Example

```

```

Relative File Paths

A relative file path points to a file relative to the current page.

In the following example, the file path points to a file in the images folder located at the root of the current web:

Example

```

```

In the following example, the file path points to a file in the images folder located in the current folder:

Example

```

```



In the following example, the file path points to a file in the images folder located in the folder one level up from the current folder:

Example

```

```

HTML The Head Element 24.

HTML **<head>** element is a container tag and is placed before the **<body>** tag. The **<head>** element contains general information about the page, meta-information, style sheet URL and document type information.

HTML **<head>** tag inside elements does not display n a body part in a web browser.

HTML **<head>** tag contains following elements that describe the metadata:

<title>, <meta>, <link>, <style>, <script>, and <base>

The HTML `<head>` element is a container for the following elements: `<title>`, `<style>`, `<meta>`, `<link>`, `<script>`, and `<base>`.

The HTML `<head>` Element

The `<head>` element is a container for metadata (data about data) and is placed between the `<html>` tag and the `<body>` tag.

HTML metadata is data about the HTML document. Metadata is not displayed.

Metadata typically define the document title, character set, styles, scripts, and other meta information.

The HTML <title> Element

The `<title>` element defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.

The `<title>` element is required in HTML documents!

The content of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

The `<title>` element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search engine-results

So, try to make the title as accurate and meaningful as possible!

A simple HTML document:

Example

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

The content of the document.....

```
</body>
</html>
```

The HTML <style> Element

The `<style>` element is used to define style information for a single HTML page:

Example

```
<style>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```

The HTML <link> Element

The `<link>` element defines the relationship between the current document and an external resource.

The `<link>` tag is most often used to link to external style sheets:

Example

```
<link rel="stylesheet" href="mystyle.css">
```



The HTML <meta> Element

The `<meta>` element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings.

The metadata will not be displayed on the page, but is used by browsers (how to display content or reload page), by search engines (keywords), and other web services.

Examples

Define the character set used:

```
<meta charset="UTF-8">
```

Define keywords for search engines:

```
<meta name="keywords" content="HTML, CSS, JavaScript">
```

Define a description of your web page:

```
<meta name="description" content="Free Web tutorials">
```



Define the author of a page:

```
<meta name="author" content="John Doe">
```

Refresh document every 30 seconds:

```
<meta http-equiv="refresh" content="30">
```

Setting the viewport to make your website look good on all devices:

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

Example of `<meta>` tags:

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

Example of `<meta>` tags:

Example

```
<meta charset="UTF-8">
<meta name="description" content="Free Web tutorials">
<meta name="keywords" content="HTML, CSS, JavaScript">
<meta name="author" content="John Doe">
```

Setting The Viewport

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

You should include the following `<meta>` element in all your web pages:

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

This gives the browser instructions on how to control the page's dimensions and scaling.

The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The `initial-scale=1.0` part sets the initial zoom level when the page is first loaded by the browser.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

HTML Layout Elements and Techniques 25.

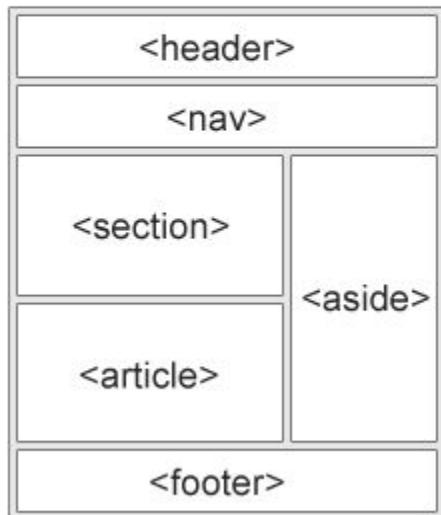


HTML layouts provide a way to arrange web pages in well-mannered, well-structured, and in responsive form or we can say that HTML layout specifies a way in which the web pages can be arranged. Web-page layout works with arrangement of visual elements of an HTML document.

Web page layout is the most important part to keep in mind while creating a website so that our website can appear professional with the great look. You can also use CSS and JAVASCRIPT based frameworks for creating layouts for responsive and dynamic website designing.

HTML Layout Elements

HTML has several semantic elements that define the different parts of a web page:



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

You can read more about semantic elements in our [HTML Semantics](#) chapter.

HTML Layout Techniques

There are four different techniques to create multicolumn layouts. Each technique has its pros and cons:

- CSS framework
- CSS float property
- CSS flexbox
- CSS grid

CSS Frameworks

If you want to create your layout fast, you can use a CSS framework, like [W3.CSS](#) or [Bootstrap](#).

CSS Float Layout

It is common to do entire web layouts using the CSS `float` property. Float is easy to learn - you just need to remember how the `float` and `clear` properties work. **Disadvantages:** Floating elements are tied to the document flow, which may harm the flexibility. Learn more about float in our [CSS Float and Clear](#) chapter.

Cities

[London](#)
[Paris](#)
[Tokyo](#)

London

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.



CSS Flexbox Layout

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

Learn more about flexbox in our [CSS Flexbox](#) chapter.

CSS Grid Layout

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

Learn more about CSS grids in our [CSS Grid View](#) chapter.

HTML Responsive Web Design 26.



Responsive web design is used to make your web page look appropriate, good, and well placed on all devices (**desktop, tablet, smartphone etc.**)

Responsive web design uses HTML and CSS to resize, hide, shrink, enlarge, or move the content. It makes the content look good on any screen.

Responsive web design is about creating web pages that look good on all devices!

A responsive web design will automatically adjust for different screen sizes and viewports.



The image shows three devices displaying a responsive website for a band. The top device is a desktop computer, the middle is a laptop, and the bottom is a smartphone. All three devices show the same basic layout of the website, demonstrating how responsive design ensures consistency across different screen sizes.

HOME BAND TOUR CONTACT MERCH

HOME BAND TOUR CONTACT MERCH

HOME

We had the best time playing at Venice Beach!

Los Angeles

We had the best time playing at Venice Beach!

THE BAND

We have created a fictional band website. *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit animi et laborum.*

THE BAND

We have created a fictional band website. *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit animi et laborum.*

THE BAND

We have created a fictional band website. *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit animi et laborum.*

Setting The Viewport

To create a responsive website, add the following `<meta>` tag to all your web pages:

Example

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



This will set the viewport of your page, which will give the browser instructions on how to control the page's dimensions and scaling.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Without the viewport meta tag:



With the viewport meta tag:



Responsive Images

Responsive images are images that scale nicely to fit any browser size.

Using the width Property

If the CSS `width` property is set to 100%, the image will be responsive and scale up and down:

Example

```

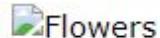
```



Show Different Images Depending on Browser Width

The HTML `<picture>` element allows you to define different images for different browser window sizes.

Resize the browser window to see how the image below change depending on the width:



Example

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
  <source srcset="flowers.jpg">
  
</picture>
```

Responsive Text Size

The text size can be set with a "vw" unit, which means the "viewport width".

That way the text size will follow the size of the browser window:



Hello World

Resize the browser window to see how the text size scales.

Example

```
<h1 style="font-size:10vw">Hello World</h1>
```

Activate
Go to Settings

Media Queries

In addition to resize text and images, it is also common to use media queries in responsive web pages.

With media queries you can define completely different styles for different browser sizes.

Example: resize the browser window to see that the three div elements below will display horizontally on large screens and stacked vertically on small screens:

Left Menu

Main Content

Right Content

Example

```
<style>
.left, .right {
  float: left;
  width: 20%; /* The width is 20%, by default */
}

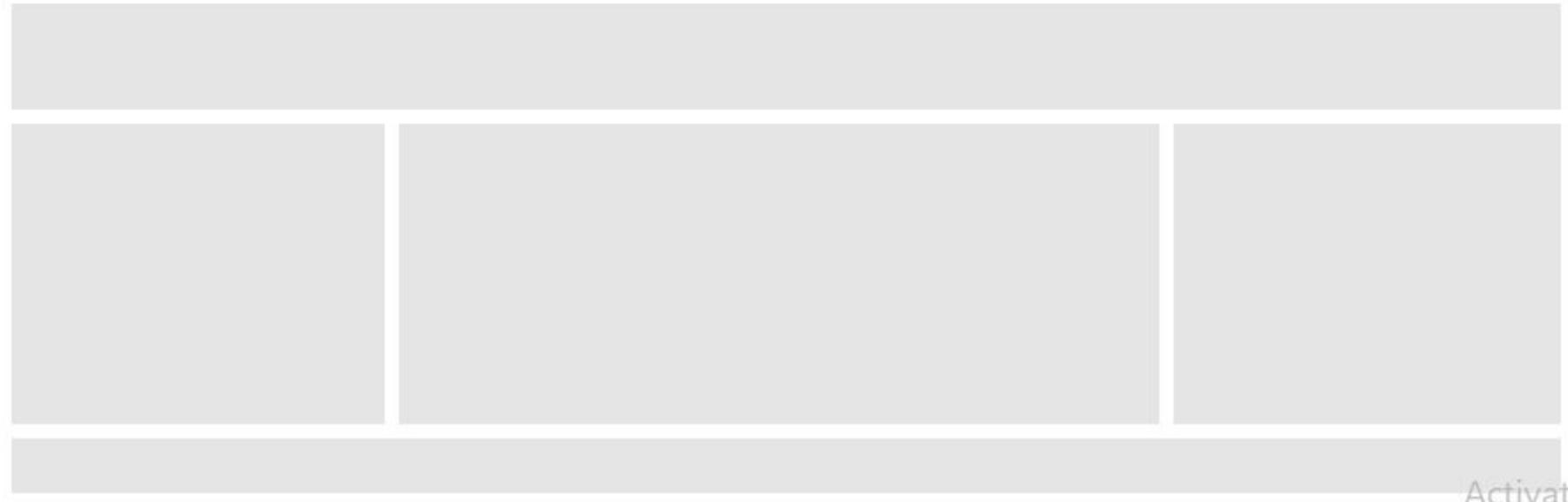
.main {
  float: left;
  width: 60%; /* The width is 60%, by default */
}

/* Use a media query to add a breakpoint at 800px: */
@media screen and (max-width: 800px) {
  .left, .main, .right {
    width: 100%; /* The width is 100%, when the viewport is 800px or smaller */
  }
}
</style>
```



Responsive Web Page - Full Example

A responsive web page should look good on large desktop screens and on small mobile phones.



Activat

W3.CSS Demo

Resize the page to see the responsiveness!

London

London is the capital city of England.

It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Paris

Paris is the capital of France.

The Paris area is one of the largest population centers in Europe, with more than 12 million inhabitants.

Tokyo

Tokyo is the capital of Japan.

It is the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Bootstrap Example</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <div class="jumbotron">
    <h1>My First Bootstrap Page</h1>
  </div>
  <div class="row">
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
      ...
    </div>
  </div>
</div>
```

HTML Semantic Elements 27.

In any language, it is essential to understand the meaning of words during communication.

And if this is a computer communication then it becomes more critical. So **HTML5** provides more semantic elements which make easy understanding of the code.

Hence Semantics defines the meaning of words and phrases, i.e.

Semantic elements= elements with a meaning. Semantic elements have a simple and clear meaning for both, the browser and the developer.

For example:

In HTML4 we have seen **<div>**, **** etc. are which are non-semantic elements. They don't tell anything about its content.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: `<div>` and `` - Tells nothing about its content.

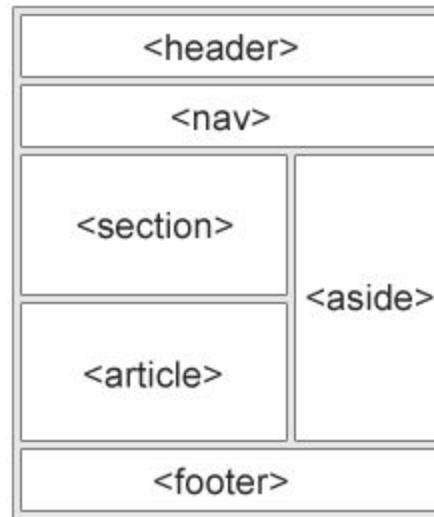
Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

Semantic Elements in HTML

Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer"> to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page:

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



HTML <section> Element

The `<section>` element defines a section in a document.

According to W3C's HTML documentation: "A section is a thematic grouping of content, typically with a heading."

A web page could normally be split into sections for introduction, content, and contact information.

Example

Two sections in a document:

```
<section>
<h1>WWF</h1>
<p>The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.</p>
</section>

<section>
<h1>WWF's Panda symbol</h1>
<p>The Panda has become the symbol of WWF. The well-known panda logo of WWF originated from a panda named Chi Chi that was transferred from the Beijing Zoo to the London Zoo in the same year of the establishment of WWF.</p>
</section>
```

HTML <article> Element

The `<article>` element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to distribute it independently from the rest of the web site.

Examples of where an `<article>` element can be used:

- Forum post
- Blog post
- Newspaper article

Example

Three articles with independent, self-contained content:

```
<article>
<h2>Google Chrome</h2>
<p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most
popular web browser today!</p>
</article>

<article>
<h2>Mozilla Firefox</h2>
<p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second
most popular web browser since January, 2018.</p>
</article>

<article>
<h2>Microsoft Edge</h2>
<p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced
Internet Explorer.</p>
</article>
```

Example 2

Use CSS to style the <article> element:

```
<html>
<head>
<style>
.all-browsers {
  margin: 0;
  padding: 5px;
  background-color: lightgray;
}

.all-browsers > h1, .browser {
  margin: 10px;
  padding: 5px;
}

.browser {
  background: white;
}

.browser > h2, p {
  margin: 4px;
  font-size: 90%;
}
```



```
margin: 4px;
font-size: 90%;
}
</style>
</head>
<body>

<article class="all-browsers">
    <h1>Most Popular Browsers</h1>
    <article class="browser">
        <h2>Google Chrome</h2>
        <p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!</p>
    </article>
    <article class="browser">
        <h2>Mozilla Firefox</h2>
        <p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018.</p>
    </article>
    <article class="browser">
        <h2>Microsoft Edge</h2>
        <p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer.</p>
    </article>
</article>
</body>
```



Activa
Go to Se

Nesting <article> in <section> or Vice Versa?

The `<article>` element specifies independent, self-contained content.

The `<section>` element defines section in a document.

Can we use the definitions to decide how to nest those elements? No, we cannot!

So, you will find HTML pages with `<section>` elements containing `<article>` elements, and `<article>` elements containing `<section>` elements.

HTML <header> Element

The `<header>` element represents a container for introductory content or a set of navigational links.

A `<header>` element typically contains:

- one or more heading elements (`<h1>` - `<h6>`)
- logo or icon
- authorship information

Note: You can have several `<header>` elements in one HTML document. However, `<header>` cannot be placed within a `<footer>`, `<address>` or another `<header>` element.

Example

A header for an <article>:

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML <footer> Element

The `<footer>` element defines a footer for a document or section.

A `<footer>` element typically contains:

- authorship information
- copyright information
- contact information
- sitemap
- back to top links
- related documents

You can have several `<footer>` elements in one document.

Example

A footer section in a document:

```
<footer>
  <p>Author: Hege Refsnes</p>
  <p><a href="mailto:hege@example.com">hege@example.com</a></p>
</footer>
```

HTML <nav> Element

The `<nav>` element defines a set of navigation links.

Notice that NOT all links of a document should be inside a `<nav>` element. The `<nav>` element is intended only for major block of navigation links.

Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

HTML <aside> Element

The `<aside>` element defines some content aside from the content it is placed in (like a sidebar).

The `<aside>` content should be indirectly related to the surrounding content.

Example

Display some content aside from the content it is placed in:

```
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing!  
I had a great summer together with my family!</p>  
  
<aside>  
  <h4>Epcot Center</h4>  
  <p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions, international  
  pavilions, award-winning fireworks and seasonal special events.</p>  
</aside>
```

Example 2

Use CSS to style the <aside> element:

```
<html>
<head>
<style>
aside {
    width: 30%;
    padding-left: 15px;
    margin-left: 15px;
    float: right;
    font-style: italic;
    background-color: lightgray;
}
</style>
</head>
<body>
```



```
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing!
I had a great summer together with my family!</p>
```

```
<aside>
<p>The Epcot center is a theme park at Walt Disney World Resort featuring exciting attractions,
international pavilions, award-winning fireworks and seasonal special events.</p>
```

Activat
Go to Set

```
<aside>
<p>The Epcot center is a theme park at Walt Disney World Resort featuring exciting attractions,
international pavilions, award-winning fireworks and seasonal special events.</p>
</aside>

<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing!
I had a great summer together with my family!</p>
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing!
I had a great summer together with my family!</p>

</body>
</html>
```

HTML <figure> and <figcaption> Elements

The `<figure>` tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

The `<figcaption>` tag defines a caption for a `<figure>` element. The `<figcaption>` element can be placed as the first or as the last child of a `<figure>` element.

The `` element defines the actual image/illustration.

Example

```
<figure>
  
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

Why Semantic Elements?

According to the W3C: "A semantic Web allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML

Below is a list of some of the semantic elements in HTML.

Semantic Elements in HTML

Below is a list of some of the semantic elements in HTML.



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

HTML Forms 28.

HTML Forms are required, when you want to collect some data from the site visitor.

For example, during user registration you would like to collect information such as **name, email address, credit card, etc.**

A form will take input from the site visitor and then will post it to a back-end application such as Python ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

Example

First name:

Last name:

The <form> Element

The HTML `<form>` element is used to create an HTML form for user input:

```
<form>
  .
  form elements
  .
</form>
```

The `<form>` element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

All the different form elements are covered in this chapter: [HTML Form Elements](#).

The <input> Element

The HTML <input> element is the most used form element.

An <input> element can be displayed in many ways, depending on the `type` attribute.

Here are some examples:



Type	Description
<input type="text">	Displays a single-line text input field
<input type="radio">	Displays a radio button (for selecting one of many choices)
<input type="checkbox">	Displays a checkbox (for selecting zero or more of many choices)
<input type="submit">	Displays a submit button (for submitting the form)
<input type="button">	Displays a clickable button

All the different input types are covered in this chapter: [HTML Input Types](#).

Activat

Go to Set

Text Fields

The `<input type="text">` defines a single-line input field for text input.

Example

A form with input fields for text:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

This is how the HTML code above will be displayed in a browser:

First name:

Last name:

Note: The form itself is not visible. Also note that the default width of an input field is 20 characters.

The <label> Element

Notice the use of the `<label>` element in the example above.

The `<label>` tag defines a label for many form elements.

The `<label>` element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

The `<label>` element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the `<label>` element, it toggles the radio button/checkbox.

The `for` attribute of the `<label>` tag should be equal to the `id` attribute of the `<input>` element to bind them together.

Radio Buttons

The `<input type="radio">` defines a radio button.

Radio buttons let a user select ONE of a limited number of choices.

Example

A form with radio buttons:

```
<p>Choose your favorite Web language:</p>

<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```

This is how the HTML code above will be displayed **in** a browser:

Choose your favorite Web language:

- HTML
 - CSS
 - JavaScript
-

Checkboxes

The `<input type="checkbox">` defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example

A form with checkboxes:

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label>
</form>
```

This is how the HTML code above will be displayed in a browser:

- I have a bike
 - I have a car
 - I have a boat
-

The Submit Button

The `<input type="submit">` defines a button for submitting the form data to a form-handler.

The form-handler is typically a file on the server with a script for processing input data.

The form-handler is specified in the form's `action` attribute.

Example

A form with a submit button:

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```



This is how the HTML code above will be displayed in a browser:

First name:

Last name:

Activate
Go to Settings

HTML Form Attributes



The Action Attribute

The `action` attribute defines the action to be performed when the form is submitted.

Usually, the form data is sent to a file on the server when the user clicks on the submit button.

In the example below, the form data is sent to a file called "action_page.php". This file contains a server-side script that handles the form data:

Example

On submit, send form data to "action_page.php":

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

Tip: If the `action` attribute is omitted, the action is set to the current page.

The Target Attribute

The `target` attribute specifies where to display the response that is received after submitting the form.

The `target` attribute can have one of the following values:

Value	Description
<code>_blank</code>	The response is displayed in a new window or tab
<code>_self</code>	The response is displayed in the current window
<code>_parent</code>	The response is displayed in the parent frame
<code>_top</code>	The response is displayed in the full body of the window
<code>framename</code>	The response is displayed in a named iframe

The default value is `_self` which means that the response will open in the current window.

Example

Here, the submitted result will open in a new browser tab:

```
<form action="/action_page.php" target="_blank">
```

The Method Attribute

The `method` attribute specifies the HTTP method to be used when submitting the form data.

The form-data can be sent as URL variables (with `method="get"`) or as HTTP post transaction (with `method="post"`).

The default HTTP method when submitting form data is GET.

Example

This example uses the GET method when submitting the form data:

```
<form action="/action_page.php" method="get">
```

Example

This example uses the POST method when submitting the form data:

```
<form action="/action_page.php" method="post">
```

Notes on GET:

- Appends the form data to the URL, in name/value pairs
- NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)
- The length of a URL is limited (2048 characters)
- Useful for form submissions where a user wants to bookmark the result
- GET is good for non-secure data, like query strings in Google

Notes on POST:

- Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

The Autocomplete Attribute

The `autocomplete` attribute specifies whether a form should have autocomplete on or off.

When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

Example

A form with autocomplete on:

```
<form action="/action_page.php" autocomplete="on">
```

The Novalidate Attribute

The `novalidate` attribute is a boolean attribute.

When present, it specifies that the form-data (`input`) should not be validated when submitted.

Example

A form with a novalidate attribute:

```
<form action="/action_page.php" novalidate>
```

HTML Form Elements

This chapter describes all the different HTML form elements.

The HTML <form> Elements

The HTML <form> element can contain one or more of the following form elements:

- <input>
- <label>
- <select>
- <textarea>
- <button>
- <fieldset>
- <legend>
- <datalist>
- <output>
- <option>
- <optgroup>



The <input> Element

One of the most used form element is the `<input>` element.

The `<input>` element can be displayed in several ways, depending on the `type` attribute.

Example

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname">
```

The <label> Element

The `<label>` element defines a label for several form elements.

The `<label>` element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

The `<label>` element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the `<label>` element, it toggles the radio button/checkbox.

The `for` attribute of the `<label>` tag should be equal to the `id` attribute of the `<input>` element to bind them together.





The <select> Element

The `<select>` element defines a drop-down list:

Example

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

The `<option>` elements defines an option that can be selected.

By default, the first item in the drop-down list is selected.

To define a pre-selected option, add the `selected` attribute to the option:

Example

```
<option value="fiat" selected>Fiat</option>
```

Visible Values:

Use the `size` attribute to specify the number of visible values:

Example

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars" size="3">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```



Allow Multiple Selections:

Use the `multiple` attribute to allow the user to select more than one value:

Example

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars" size="4" multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

The <textarea> Element

The <textarea> element defines a multi-line input field (a text area):

Example

```
<textarea name="message" rows="10" cols="30">  
The cat was playing in the garden.  
</textarea>
```

The `rows` attribute specifies the visible number of lines in a text area.

The `cols` attribute specifies the visible width of a text area.

This is how the HTML code above will be displayed in a browser:

The cat was playing in the garden.

You can also define the size of the text area by using CSS:

Example

```
<textarea name="message" style="width:200px; height:600px;">  
The cat was playing in the garden.  
</textarea>
```

The <button> Element

The `<button>` element defines a clickable button:

Example

```
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
```

The <fieldset> and <legend> Elements

The `<fieldset>` element is used to group related data in a form.

The `<legend>` element defines a caption for the `<fieldset>` element.

Example

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personalia:</legend>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" value="John"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" value="Doe"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

This is how the HTML code above will be displayed in a browser:

Personalia:

First name:

John

Last name:

Doe

Submit

The <datalist> Element

The `<datalist>` element specifies a list of pre-defined options for an `<input>` element.

Users will see a drop-down list of the pre-defined options as they input data.

The `list` attribute of the `<input>` element, must refer to the `id` attribute of the `<datalist>` element.

Example

```
<form action="/action_page.php">
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

This chapter describes the different types for the HTML `<input>` element.

HTML Input Types

Here are the different input types you can use in HTML:

- `<input type="button">`
- `<input type="checkbox">`
- `<input type="color">`
- `<input type="date">`
- `<input type="datetime-local">`
- `<input type="email">`
- `<input type="file">`
- `<input type="hidden">`
- `<input type="image">`
- `<input type="month">`

- <input type="number">
- <input type="password">
- <input type="radio">
- <input type="range">
- <input type="reset">
- <input type="search">
- <input type="submit">
- <input type="tel">
- <input type="text">
- <input type="time">
- <input type="url">
- <input type="week">

Tip: The default value of the `type` attribute is "text".

Input Type Text

`<input type="text">` defines a **single-line text input field**:

Example

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

This is how the HTML code above will be displayed in a browser:

First name:

Last name:

Input Type Password

`<input type="password">` defines a **password field**:

Example

```
<form>
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username"><br>
  <label for="pwd">Password:</label><br>
  <input type="password" id="pwd" name="pwd">
</form>
```

This is how the HTML code above will be displayed in a browser:

Username:

Password:

Input Type Password

`<input type="password">` defines a **password field**:

Example

```
<form>
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username"><br>
  <label for="pwd">Password:</label><br>
  <input type="password" id="pwd" name="pwd">
</form>
```

Input Type Submit

`<input type="submit">` defines a button for **submitting** form data to a **form-handler**.

The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's `action` attribute:

Example

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```



This is how the HTML code above will be displayed in a browser:

First name:

Last name:

If you omit the submit button's value attribute, the button will get a default text:

Example

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit">
</form>
```



Input Type Reset

`<input type="reset">` defines a **reset button** that will reset all form values to their default values:

Example

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```



This is how the HTML code above will be displayed in a browser:

First name:

Last name:



Input Type Radio

`<input type="radio">` defines a **radio button**.

Radio buttons let a user select ONLY ONE of a limited number of choices:

Example

```
<p>Choose your favorite Web language:</p>

<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```



Input Type Checkbox

`<input type="checkbox">` defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label>
</form>
```

Input Type Color

The `<input type="color">` is used for input fields that should contain a color.

Depending on browser support, a color picker can show up in the input field.

Example

```
<form>
  <label for="favcolor">Select your favorite color:</label>
  <input type="color" id="favcolor" name="favcolor">
</form>
```

Input Type Button

<input type="button"> defines a **button**:

Example

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```

Input Type Date

The `<input type="date">` is used for input fields that should contain a date.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  <label for="birthday">Birthday:</label>
  <input type="date" id="birthday" name="birthday">
</form>
```

You can also use the `min` and `max` attributes to add restrictions to dates:

Example

```
<form>
  <label for="datemax">Enter a date before 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>
  <label for="datemin">Enter a date after 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02">
</form>
```

Input Type Email

The `<input type="email">` is used for input fields that should contain an e-mail address.

Depending on browser support, the e-mail address can be automatically validated when submitted.

Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

Example

```
<form>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email">
</form>
```

Input Type Image

The `<input type="image">` defines an image as a submit button.

The path to the image is specified in the `src` attribute.

Example

```
<form>
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
</form>
```

Input Type File

The `<input type="file">` defines a file-select field and a "Browse" button for file uploads.

Example

```
<form>
  <label for="myfile">Select a file:</label>
  <input type="file" id="myfile" name="myfile">
</form>
```

Input Type Hidden

The `<input type="hidden">` defines a hidden input field (not visible to a user).

A hidden field lets web developers include data that cannot be seen or modified by users when a form is submitted.

A hidden field often stores what database record that needs to be updated when the form is submitted.

Note: While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!

Example

```
<form>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="hidden" id="custId" name="custId" value="3487">
  <input type="submit" value="Submit">
</form>
```

Input Type Number

The `<input type="number">` defines a **numeric** input field.

You can also set restrictions on what numbers are accepted.

The following example displays a numeric input field, where you can enter a value from 1 to 5:

Example

```
<form>
  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

Attribute	Description
checked	Specifies that an input field should be pre-selected when the page loads (for type="checkbox" or type="radio")
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

Input Type Search

The `<input type="search">` is used for search fields (a search field behaves like a regular text field).

Example

```
<form>
  <label for="gsearch">Search Google:</label>
  <input type="search" id="gsearch" name="gsearch">
</form>
```

Input Type Tel

The `<input type="tel">` is used for input fields that should contain a telephone number.

Example

```
<form>
  <label for="phone">Enter your phone number:</label>
  <input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}>
</form>
```

Input Type Time

The `<input type="time">` allows the user to select a time (no time zone).

Depending on browser support, a time picker can show up in the input field.

Example

```
<form>
  <label for="appt">Select a time:</label>
  <input type="time" id="appt" name="appt">
</form>
```



Input Type Url

The `<input type="url">` is used for input fields that should contain a URL address.

Depending on browser support, the url field can be automatically validated when submitted.

Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

Example

```
<form>
  <label for="homepage">Add your homepage:</label>
  <input type="url" id="homepage" name="homepage">
</form>
```



HTML Input Attributes

The value Attribute

The `input` `value` attribute specifies an initial value for an input field:

Example

Input fields with initial (default) values:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

The readonly Attribute

The input `readonly` attribute specifies that an input field is read-only.

A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it).

The value of a read-only input field will be sent when submitting the form!

Example

A read-only input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" readonly><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

The disabled Attribute

The `input disabled` attribute specifies that an input field should be disabled.

A disabled input field is unusable and un-clickable.

The value of a disabled input field will not be sent when submitting the form!

Example

A disabled input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

The size Attribute

The input `size` attribute specifies the visible width, in characters, of an `input` field.

The default value for `size` is 20.

Note: The `size` attribute works with the following input types: `text`, `search`, `tel`, `url`, `email`, and `password`.

Example

Set a width for an input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="text" id="pin" name="pin" size="4">
</form>
```

The maxlength Attribute

The input `maxlength` attribute specifies the maximum number of characters allowed in an input field.

Note: When a `maxlength` is set, the input field will not accept more than the specified number of characters. However, this attribute does not provide any feedback. So, if you want to alert the user, you must write JavaScript code.

Example

Set a maximum length for an input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="text" id="pin" name="pin" maxlength="4" size="4">
</form>
```

The min and max Attributes



The input `min` and `max` attributes specify the minimum and maximum values for an input field.

The `min` and `max` attributes work with the following input types: number, range, date, datetime-local, month, time and week.

Tip: Use the `max` and `min` attributes together to create a range of legal values.

Example

Set a max date, a min date, and a range of legal values:

```
<form>
  <label for="datemax">Enter a date before 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>

  <label for="datemin">Enter a date after 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02"><br><br>

  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

The multiple Attribute

The input `multiple` attribute specifies that the user is allowed to enter more than one value in an `input` field.

The `multiple` attribute works with the following input types: `email`, and `file`.

Example

A file upload field that accepts multiple values:

```
<form>
  <label for="files">Select files:</label>
  <input type="file" id="files" name="files" multiple>
</form>
```



The pattern Attribute

The `input pattern` attribute specifies a regular expression that the input field's value is checked against, when the form is submitted.

The `pattern` attribute works with the following input types: text, date, search, url, tel, email, and password.

Tip: Use the global `title` attribute to describe the pattern to help the user.

Tip: Learn more about [regular expressions](#) in our JavaScript tutorial.

Example

An input field that can contain only three letters (no numbers or special characters):

```
<form>
  <label for="country_code">Country code:</label>
  <input type="text" id="country_code" name="country_code"
    pattern="[A-Za-z]{3}" title="Three letter country code">
</form>
```

The placeholder Attribute

The `input placeholder` attribute specifies a short hint that describes the expected value of an input field (a sample value or a short description of the expected format).

The short hint is displayed in the `input` field before the user enters a value.

The `placeholder` attribute works with the following `input` types: `text`, `search`, `url`, `tel`, `email`, and `password`.

Example

An `input` field with a placeholder text:

```
<form>
  <label for="phone">Enter a phone number:</label>
  <input type="tel" id="phone" name="phone"
placeholder="123-45-678"
pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```



The required Attribute

The input `required` attribute specifies that an input field must be filled out before submitting the form.

The `required` attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Example

A required input field:

```
<form>
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>
</form>
```

The step Attribute

The `input step` attribute specifies the legal number intervals for an input field.

Example: if `step="3"`, legal numbers could be -3, 0, 3, 6, etc.

Tip: This attribute can be used together with the `max` and `min` attributes to create a range of legal values.

The `step` attribute works with the following input types: `number`, `range`, `date`, `datetime-local`, `month`, `time` and `week`.

Example

An input field with a specified legal number intervals:

```
<form>
  <label for="points">Points:</label>
  <input type="number" id="points" name="points" step="3">
</form>
```

The autofocus Attribute

The input `autofocus` attribute specifies that an input field should automatically get focus when the page loads.

Example

Let the "First name" input field automatically get focus when the page loads:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" autofocus><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

The height and width Attributes

The input `height` and `width` attributes specify the height and width of an `<input type="image">` element.

Tip: Always specify both the height and width attributes for images. If height and width are set, the space required for the image is reserved when the page is loaded. Without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the images load).

Example

Define an image as the submit button, with height and width attributes:

```
<form>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
</form>
```

The list Attribute

The input `list` attribute refers to a `<datalist>` element that contains pre-defined options for an `<input>` element.

Example

An `<input>` element with pre-defined values in a `<datalist>`:

```
<form>
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

The autocomplete Attribute

The input `autocomplete` attribute specifies whether a form or an input field should have autocomplete on or off.

Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values.

The `autocomplete` attribute works with `<form>` and the following `<input>` types: text, search, url, tel, email, password, datepickers, range, and color.

Example

An HTML form with autocomplete on, and off for one input field:

```
<form action="/action_page.php" autocomplete="on">
    <label for="fname">First name:</label>
    <input type="text" id="fname" name="fname"><br><br>
    <label for="lname">Last name:</label>
    <input type="text" id="lname" name="lname"><br><br>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" autocomplete="off"><br><br>
    <input type="submit" value="Submit">
</form>
```

The form Attribute

The input `form` attribute specifies the form the `<input>` element belongs to.

The value of this attribute must be equal to the `id` attribute of the `<form>` element it belongs to.

Example

An input field located outside of the HTML form (but still a part of the form):

```
<form action="/action_page.php" id="form1">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
</form>

<label for="lname">Last name:</label>
<input type="text" id="lname" name="lname" form="form1">
```

The formaction Attribute

The input `formaction` attribute specifies the URL of the file that will process the input when the form is submitted.

Note: This attribute overrides the `action` attribute of the `<form>` element.

The `formaction` attribute works with the following input types: submit and image.

Example

An HTML form with two submit buttons, with different actions:

```
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formaction="/action_page2.php" value="Submit as Admin">
</form>
```

The `formenctype` Attribute

The input `formenctype` attribute specifies how the form-data should be encoded when submitted (only for forms with `method="post"`).

Note: This attribute overrides the `enctype` attribute of the `<form>` element.

The `formenctype` attribute works with the following input types: submit and image.

Example

A form with two submit buttons. The first sends the form-data with default encoding, the second sends the form-data encoded as "multipart/form-data":

```
<form action="/action_page_binary.asp" method="post">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
        value="Submit as Multipart/form-data">
</form>
```

The formmethod Attribute

The input `formmethod` attribute defines the HTTP method for sending form-data to the action URL.

Note: This attribute overrides the `method` attribute of the `<form>` element.

The `formmethod` attribute works with the following input types: `submit` and `image`.

The form-data can be sent as URL variables (`method="get"`) or as an HTTP post transaction (`method="post"`).

Notes on the "get" method:

- This method appends the form-data to the URL in name/value pairs
- This method is useful for form submissions where a user want to bookmark the result
- There is a limit to how much data you can place in a URL (varies between browsers), therefore, you cannot be sure that all of the form-data will be correctly transferred
- Never use the "get" method to pass sensitive information! (password or other sensitive information will be visible in the browser's address bar)

Notes on the "post" method:

- This method sends the form-data as an HTTP post transaction
- Form submissions with the "post" method cannot be bookmarked
- The "post" method is more robust and secure than "get", and "post" does not have size limitations

Example

A form with two submit buttons. The first sends the form-data with method="get". The second sends the form-data with method="post":

```
<form action="/action_page.php" method="get">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit using GET">
  <input type="submit" formmethod="post" value="Submit using POST">
</form>
```

The formtarget Attribute

The input `formtarget` attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

Note: This attribute overrides the target attribute of the `<form>` element.

The `formtarget` attribute works with the following input types: submit and image.

Example

A form with two submit buttons, with different target windows:

```
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formtarget="_blank" value="Submit to a new window/tab">
</form>
```

The formnovalidate Attribute

The input `formnovalidate` attribute specifies that an `<input>` element should not be validated when submitted.

Note: This attribute overrides the `novalidate` attribute of the `<form>` element.

The `formnovalidate` attribute works with the following input types: submit.

Example

A form with two submit buttons (with and without validation):

```
<form action="/action_page.php">
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formnovalidate="formnovalidate"
        value="Submit without validation">
</form>
```

The novalidate Attribute

The `novalidate` attribute is a `<form>` attribute.

When present, `novalidate` specifies that all of the form-data should not be validated when submitted.

Example

Specify that no form-data should be validated on submit:

```
<form action="/action_page.php" novalidate>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email"><br><br>
  <input type="submit" value="Submit">
</form>
```

HTML Video section 29.

HTML5 supports <video> tag also. The HTML video tag is used for streaming video files such as a movie clip, song clip on the web page.

Currently, there are three video formats supported for HTML video tag:

1. mp4
2. webM
3. ogg

The HTML <video> Element

To show a video in HTML, use the `<video>` element:

Example

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

How it Works

The `controls` attribute adds video controls, like play, pause, and volume.

It is a good idea to always include `width` and `height` attributes. If height and width are not set, the page might flicker while the video loads.

The `<source>` element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

The text between the `<video>` and `</video>` tags will only be displayed in browsers that do not support the `<video>` element.

HTML <video> Autoplay

To start a video automatically, use the `autoplay` attribute:

Example

```
<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```



Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add `muted` after `autoplay` to let your video start playing automatically (but muted):

Example

```
<video width="320" height="240" autoplay muted>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

Browser Support

The numbers in the table specify the first browser version that fully supports the `<video>` element.

Element					
<code><video></code>	4.0	9.0	3.5	4.0	10.5

HTML Video Formats

There are three supported video formats: MP4, WebM, and Ogg. The browser support for the different formats is:

Browser	MP4	WebM	Ogg
Edge	YES	YES	YES
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

HTML Video - Media Types

File Format	Media Type
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

HTML Video - Methods, Properties, and Events

The HTML DOM defines methods, properties, and events for the `<video>` element.

This allows you to load, play, and pause videos, as well as setting duration and volume.

There are also DOM events that can notify you when a video begins to play, is paused, etc.

HTML Video Tags

Tag	Description
<u><video></u>	Defines a video or movie
<u><source></u>	Defines multiple media resources for media elements, such as <video> and <audio>
<u><track></u>	Defines text tracks in media players

HTML Audio section 30.

Html audio tag is used to define sounds such as music and other audio clips. Currently there are three supported file format for HTML5 audio tag.

1. mp3
2. wav
3. ogg

The HTML `<audio>` element is used to play an audio file on a web page.

The HTML `<audio>` Element

To play an audio file in HTML, use the `<audio>` element:

Example

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

HTML Audio - How It Works

The `controls` attribute adds audio controls, like play, pause, and volume.

The `<source>` element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the `<audio>` and `</audio>` tags will only be displayed in browsers that do not support the `<audio>` element.

HTML `<audio>` Autoplay

To start an audio file automatically, use the `autoplay` attribute:

Example

```
<audio controls autoplay>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add `muted` after `autoplay` to let your audio file start playing automatically (but muted):

Example

```
<audio controls autoplay muted>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

HTML Audio Formats

There are three supported audio formats: MP3, WAV, and OGG. The browser support for the different formats is:

Browser	MP3	WAV	OGG
Edge/IE	YES	YES*	YES*
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

*From Edge 79

HTML Audio - Media Types

File Format	Media Type
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav



HTML Audio - Methods, Properties, and Events

The HTML DOM defines methods, properties, and events for the `<audio>` element.

This allows you to load, play, and pause audios, as well as set duration and volume.

There are also DOM events that can notify you when an audio begins to play, is paused, etc.

For a full DOM reference, go to our [HTML Audio/Video DOM Reference](#).



HTML Audio Tags

Tag	Description
<code><audio></code>	Defines sound content
<code><source></code>	Defines multiple media resources for media elements, such as <code><video></code> and <code><audio></code>

HTML YouTube Videos Section 32.

In the early days, adding a video to a webpage was a real challenge since one had to convert the videos to different formats to make them play in all browsers.

Converting videos to different formats can be difficult and time-consuming.

Now, adding a video to a web page has become as easy as copying and pasting and a very apt solution to add videos to a website is using Youtube. Youtube helps to host a video for a user so that they can be further embedded on web pages.

YouTube displays an id like “BGAk3_2zi8k”, whenever a video is saved or played. This id is further used as a referral for the youtube video to be embedded in the webpage.

Struggling with Video Formats?

Converting **videos** to different formats can be difficult and time-consuming.

An easier solution is to **let YouTube play the videos in your web page**.

YouTube Video Id

YouTube will display an **id** (like tgbNymZ7vqY), when you save (or play) a video.

You can **use this id**, and refer to your video in the **HTML code**.

Playing a YouTube Video in HTML

To play your video on a web page, do the following:

- Upload the video to YouTube
- Take a **note** of the video **id**
- Define an **<iframe>** element in your web page
- Let the **src** attribute point to the video URL
- Use the **width** and **height** attributes to specify the dimension of the player
- Add any other parameters to the URL (see below)

Example

```
<iframe width="420" height="315"  
src="https://www.youtube.com/embed/tgbNymZ7vqY">  
</iframe>
```

YouTube Autoplay + Mute

You can let your video start playing automatically when a user visits the page, by adding `autoplay=1` to the YouTube URL. However, automatically starting a video is annoying for your visitors!

Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add `mute=1` after `autoplay=1` to let your video start playing automatically (but muted).

YouTube - Autoplay + Muted

```
<iframe width="420" height="315"  
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1">  
</iframe>
```

YouTube Playlist

A comma separated list of videos to play (in addition to the original URL).

YouTube Loop

Add `loop=1` to let your video loop forever.

Value 0 (default): The video will play only once.

Value 1: The video will loop (forever).



YouTube - Loop

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1">
</iframe>
```

YouTube Controls

Add `controls=0` to not display controls in the video player.

Value 0: Player controls does not display.

Value 1 (default): Player controls display.

YouTube - Controls

```
<iframe width="420" height="315"  
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">  
</iframe>
```

HTML Projects

Documentation Page

News Website

Tribute page

Personal Portfolio Page

Survey form

Music Store Page

Landing Page

Resume Interactive

Event Page

Create an Email Newsletter

Build a Form

Website Products Display

Interactive Restaurant Website