HTML | a beginner's guide

HTML (– **H**yper **T**ext **M**arkup **L**anguage) is the markup language every web developer needs to know and it is the basic for all the contents on a website.

Table of Contents:

1 Introduction

2 Syntax and Text Formatting

2.1 General Syntax of HTML elements

2.2 Paragraphs

2.3 Headlines

2.4 Bold and Italics

2.5 Lists

2.5.1 Unordered List

2.5.2 Ordered List

2.5.3 Definition List

2.6 Quotations and Citations

2.6.1 <blockquote> for Quotations

2.6.2 <q> for Short Quotations

2.6.3 <abbr> for Abbreviations

2.6.4 <address> for Contact Information

2.6.5 <cite> for Work Title

2.6.6 <bdo for Bi-Directional Override

2.7 code, pre, and br Elements

2.8 Superscript, Subscript and Small text

2.9 Inline and Block-level Elements

3 Power and Features of HTML

3.1 Debugging HTML

3.2 HTML Attributes

3.3 Commenting in HTML and its Evolution

3.4 Character Entity

4 Linking and Navigation

4.1 Links

4.2 URL Paths

4.3 Navigations

5 Images and Graphics

5.1 Images

5.2 Supporting Image Formats

5.3 Responsible and Responsive Images

5.3.1 Responsible Images

5.3.2 Responsible Width

5.3.3 Responsive Pictures

5.4 Figure and Figcaption

6 Media in HTML

6.1 Audio

6.2 Video

6.3 Captions and Subtitles

6.4 Embedding Media through <iframes>

7 Identifying HTML Content and its Structure

7.1 Supporting Languages

7.2 Generic Elements <div> and <span>

7.3 The Structure of HTML Page

7.4 Document Head

7.5 Structural Elements of HTML

8 Forms and Interactive Elements

8.1 Basics of HTML Form

8.2 Additional Elements in Form

9 Structuring Tabular Data

9.1 When to use Tables

9.2 Building HTML Tables

1 Introduction

HTML provides the conduit for content, words, images, video, audio, and forms, it's a solid foundation on which everything is built. HTML doesn't contain any programming logic, or loops or functions. Instead, it's all vocabulary or declarations and has some simple codes which declares, "Hey, make this a paragraph!", "Hey, make this a heading!", "Hey, show this as a code", so it's also called as a 'declarative language'. HTML markup is like a text document, but it has a extension of '.html' and it doesn't need any complier to run like other programming languages. It runs or opens on any browser which supports HTML.

2 Syntax and Text Formatting

The structure of a HTML document is controlled by some simple HTML tags known as elements, and in a HTML document each and every content is wrapped inside its own kind of element, and they have their own syntax.

2.1 General Syntax of HTML Elements

The HTML element contains of both tags and the content which is to be displayed to the user who views the website. HTML tag is which looks like by starting out with a less-than (**<**) symbol, followed by the correct letter, or word, or an abbreviated word and finishing with a greater-than (**>**) symbol and is known as opening tag.

(e.g.): *<p>* - 'p' tag which is used for defining some content as a 'paragraph'

<a> - 'a' tag which is used for making some content as a 'clickable link'

<h1> - 'h1' tag which is used for showing some content as a 'heading'

There are actually two kinds of tags, opening and closing tags. The closing tag is very similar to the opening tag but it has a forward slash (**/**) next to the less-than symbol.

(e.g.): *</p> , </a>, </h1>*

Elements can be nested inside one another, and the elements nested inside the other is considered as their child elements and the element which wraps over every child is considered as their parent. In fact, the entire HTML document is a whole bunch of elements nested inside of each other.

(e.g.): *</p>*This is a parent element containing it's *<im>*own child*</im>* nested inside it*</p >*

Things to remember, everything that comes after the closing tag is will be displayed on the same HTML page, but it will be understood by the browser to be outside of the element.

2.2 Paragraphs

To markup a paragraph properly you should use an opening '**p**' tag at the beginning of the paragraph, and then a closing '**p**' tag at the end. And whatever that comes between these tags is marked as paragraphs. If you forget to put the closing tag everything that comes after the opening tag is considered as paragraph by the browser.

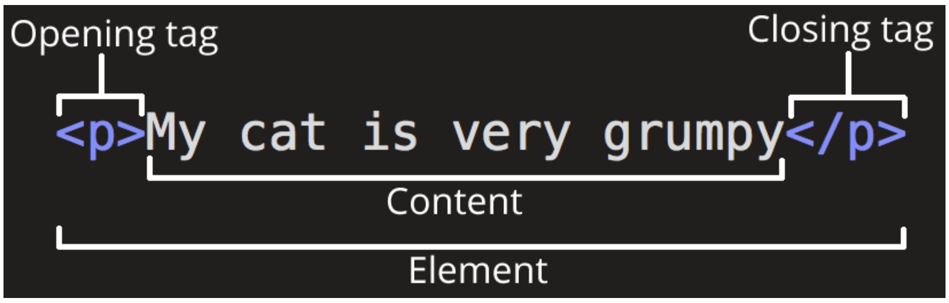


Figure 1. 1 – Syntax of HTML elements with both tags

As you can see in the (Figure 1.1) the passage or paragraph between the opening and closing tags is known as content, and the whole thing which includes the content, opening and closing tags is called an element. But not every element has a closing tag.

2.3 Headlines

As you can see in this guide, there are titles, headlines and sub-headlines highlighted or bolded to make contents easier to read, locate and understand. Like this, webpages often have a lot of titles, headlines and sub-headlines as well, and which helps people easier to understand the structure of what's on the page.

There are some HTML elements to marking up headlines in bold and in different sizes, for that there are six headline elements which are '**h1**', '**h2**', '**h3**', '**h4**', '**h5**', '**h6**'.

(e.g.): *<h1>*Headline marked up with h1*</h1>*
*<h2>*Headline marked up with h2*</h2>*
*<h3>*Headline marked up with h3*</h3>*
*<h4>*Headline marked up with h4*</h4>*
*<h5>*Headline marked up with h5*</h5>*
*<h6>*Headline marked up with h6*</h6>*

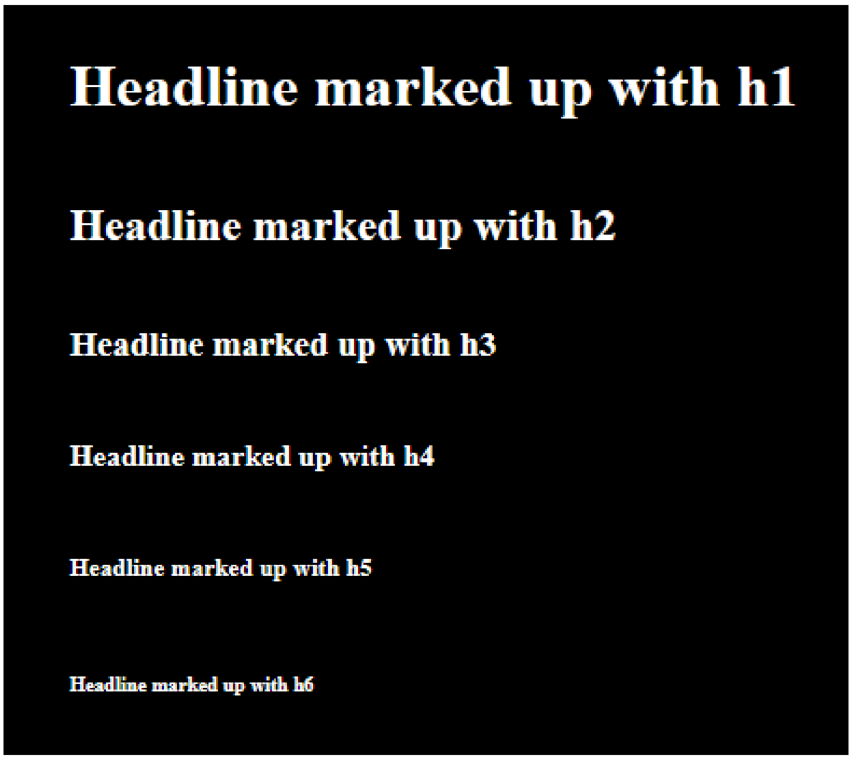

Headline marked up with h1
Headline marked up with h2
Headline marked up with h3
Headline marked up with h4
Headline marked up with h5
Headline marked up with h6

Figure 1. 2 – Six different headline elements

As you can see in the (Figure 1.2) which displays the different markup of headlines elements in various sizes according to their in-built styles. (*click here for live example*)

2.4 Bold and Italics

There are four HTML elements under Bold and Italics, two for bold and two for italics. While italicizing you can use ‘**i**’ and ‘**em**’ elements. (*click here for live example*)

‘*<i>*’ – is used to apply only visual italics to some text.

‘*<em>’* – is used to put emphasis on some text.

(e.g.): *<i>*Visually italicized text*</i>*

*<em>*Emphasized text*</em>*

Like italics, there are also two elements you can use to mark something as bold. Which is ‘**strong**’ and ‘**b**’ elements. (*click here for live example*)

‘*<strong>*’ – is used convey seriousness, or importance, or urgency like emphasis

‘*<i>*’ – is used to apply visually styling some text as bold

(e.g.): *<strong>*Important Disclaimer*</strong>*

*<b>*Visually bolded text*</b>*

2.5 Lists

HTML Lists are used to specify lists of information. All lists may contain one or more list elements. There are three different types of HTML lists

- Unordered List
- Ordered List
- Definition List

2.5.1 Unordered List

Unordered list is used the most often. Each item inside this list is nested inside ‘**li’** element which is the child elements of unordered list. And the element used to make an unordered list is ‘**ul**’. (*click here for live example*)

Below is one of the examples for unordered list of grocery items:

(e.g.): *<ul>*

*<li>*Sugar*</li>*

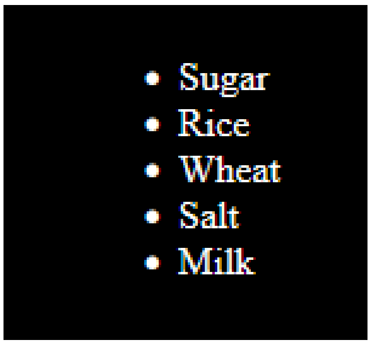
*<li>*Rice*</li>*

*<li>*Wheat*</li>*

*<li>*Salt*</li>*

*<li>*Milk*</li>*

</ul>



- Sugar
- Rice
- Wheat
- Salt
- Milk

Figure 1. 3 Unordered List

2.5.2 Ordered List

Ordered list are very similar to the unordered list and here the items in the list are ordered by natural numbers, and the element used to make an ordered list is '**ol**'. (*click here for live example*)

(e.g.) *<ol>*

*<li>*Sugar*</li>*

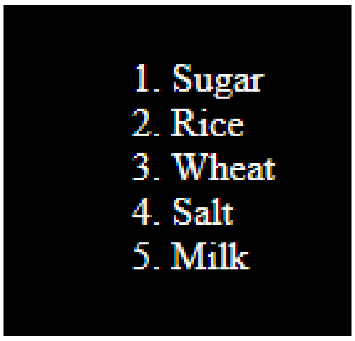
*<li>*Rice*</li>*

*<li>*Wheat*</li>*

*<li>*Salt*</li>*

*<li>*Milk*</li>*

</ol>



1. Sugar
2. Rice
3. Wheat
4. Salt
5. Milk

Figure 1. 4 Ordered List

2.5.3 Definition List

The HTML Definition list or Description list displays elements in definition form like in dictionary. There are three tags used to define definition list which is '**dl**', '**dt**', '**dd**'.

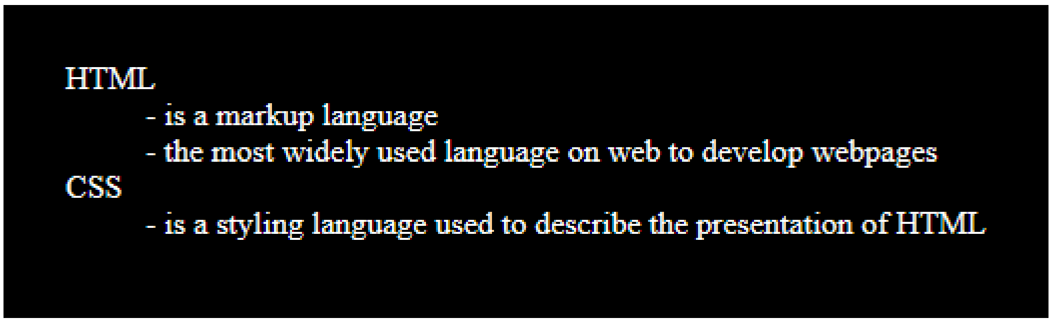
<dl> - defines the description or definition list.

<dt> - defines the data term.

<dd> - defines the description list.

(e.g.) (*click here for live example*)

```
<dl>
    <dt>HTML</dt>
        <dd>is a markup language</dd>
        <dd>the most widely used language on web to develop webpages</dd>
    <dt>CSS</dt>
        <dd>is a styling language used to describe the presentation of HTML</dd>
</dl>
```



```
HTML
        - is a markup language
        - the most widely used language on web to develop webpages
CSS
        - is a styling language used to describe the presentation of HTML
```

Figure 1. 5 Description List

2.6 Quotations and Citations

There are some HTML elements used for special purpose, and they are ‘**blockquote**’, ‘**q**’, ‘**abbr**’, ‘**address**’, ‘**cite**’ and ‘**bdo**’

2.6.1 <blockquote> for Quotations

The HTML ***’blockquote’*** element defines a section that is quoted from another source, and usually blockquote elements are indented by browsers. It is often used to mention someone’s feedback or comment or words. (*click here for live example*)

(e.g.)
```
<p>Here is a quote from WWF's website:</p>
<blockquote cite="http://www.worldwildlife.org/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>
```

2.6.2 <q> for Short Quotations

The HTML '**q**' element defines a short quotation, and the browser inserts quotations around the quotes instead of indentation in the beginning. (*click here for live example*)

(e.g.) <p>Title of this document: <q>HTML Guide for Beginners</q></p>

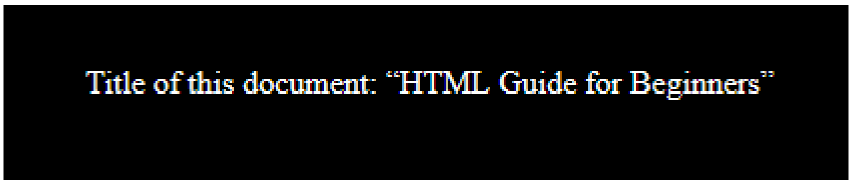

Title of this document: "HTML Guide for Beginners"

Figure 1. 6 Output of <q> tag

2.6.3 <abbr> for Abbreviations

The HTML '**abbr**' element defines an abbreviation or an acronym, like "HTML", "CSS", "WHO", "ATM". It has an additional benefit of showing the description for an abbreviation when you hover over the element, and it can be achieved by a global attribute 'title'. (*click here for live example*)

(e.g.) *<p><abbr title="Hyper Text Markup Language">*HTML*</abbr>* is used to create static web pages and web applications. *<abbr title="Cascading Style Sheets">*CSS*</abbr>* is responsible for the presentation of documents written in a markup language.*</p>*

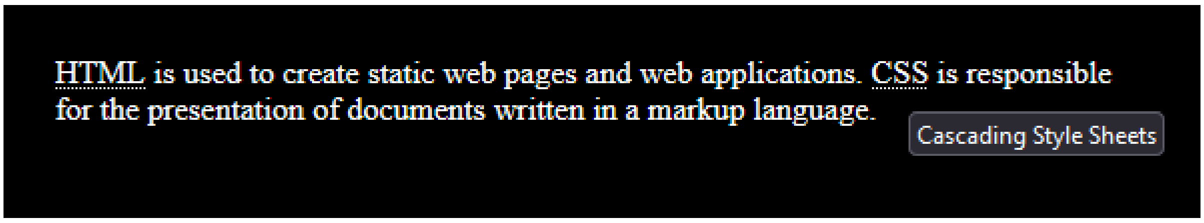

HTML is used to create static web pages and web applications. CSS is responsible for the presentation of documents written in a markup language.

Figure 1. 7 Output of <abbr> tag

2.6.4 <address> for Contact Information

The HTML '**address**' element 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 inside the element usually renders in *italic,* and browsers will always add a line break before and after the element. (click here for live example)

(e.g.) *<address>*
Written by Peter<br>
Visit us at:<br>
webdeveloper.com
</address>

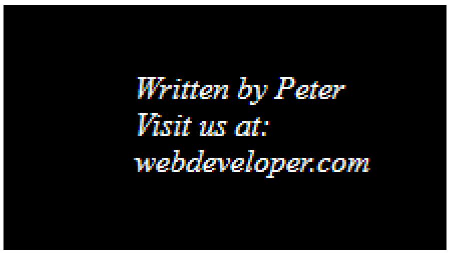

Written by Peter
Visit us at:
webdeveloper.com

Figure 1. 8 Output of <address> tag

2.6.5 <cite> for Work Title

The HTML '**cite**' element defines the title of a creative work like a book, or a poem, or a song, or a movie, or a painting, or a sculpture, etc. The text inside this element also renders in *italic*. (click here for live example)

(e.g.) *<p>*Richard Russell Riordan Jr. is an American author. He is known for writing the *<cite>*Percy Jackson*</cite>* & the *<cite>*Olympians*</cite>*.*<p>*

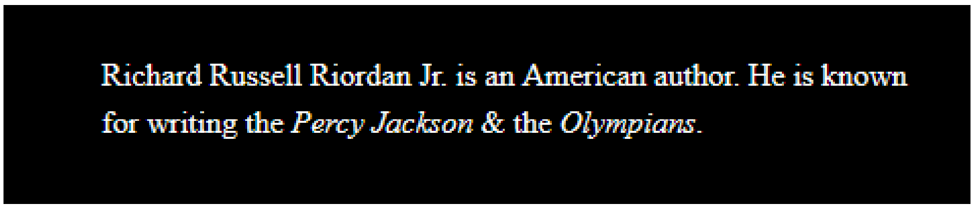

Richard Russell Riordan Jr. is an American author. He is known for writing the *Percy Jackson* & the *Olympians*.

Figure 1. 9 Output of <cite> tag

2.6.6 <bdo> for Bi-Directional Override

The HTML '**bdo**' element is used to override the current direction of text. And the text direction is changed using the global attribute 'dir'. (click here for live example)

(e.g.) *<bdo dir="rtl">*This text will be written from right to left*</bdo>*

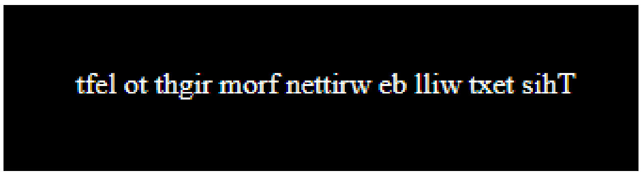

tfel ot thgir morf nettirw eb lliw txet sihT

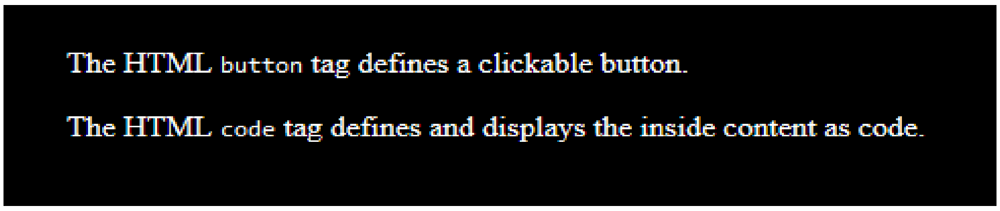
Figure 1. 10 Output of <bdo> tag

2.7 code, pre, br Elements

The HTML ‘**code**’ element is used to display a piece of computer code as code by using the browser’s default monospace font. (*click here for live example*)

(e.g.) *<p>*The HTML *<code>*button*</code>* tag defines a clickable button*</p>*

*<p>*The HTML *<code>*code*</code>* tag defines and displays the inside content as code.*</p>*



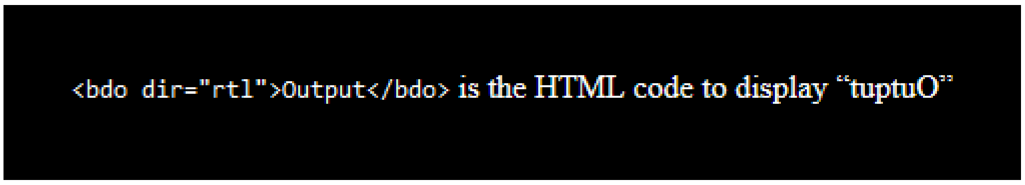
The HTML `button` tag defines a clickable button.

The HTML `code` tag defines and displays the inside content as code.

Figure 1. 11 Output of <code> tag

NOTE: If you want to display a HTML code using ‘**code**’ element, the lesser than and greater than symbols should be replace with HTML entities ‘**&lt;**’ and ‘**&gt;**’ (*see working example*)

(e.g.) *<p><code>*&lt;bdo dir="rtl"&gt;Output&lt;/bdo&gt;*</code>* is the HTML code to display *<q>*tuptuO*</q></p>*



`<bdo dir="rtl">Output</bdo>` is the HTML code to display “tuptuO”

Figure 1. 12 Output of <code> tag used to display HTML code

Usually, browser doesn’t care about the white spaces are indentation in the HTML document, but what if you want to show the white spaces and indentation along with the code. So, there comes the HTML ‘**pre**’ and ‘**br**’.

The HTML ‘**br**’ element adds line break, which helps to end the current line and start a new line. It has no closing tag and denoted by *<br>*. (*click here for live example*)

(e.g.) *<p>*This is a sample paragraph which breaks here*<br>*
and starts a new line here and can be breaks*<br>*
anywhere according to the purpose*</p>*

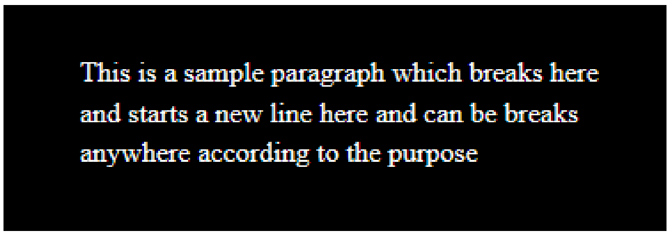
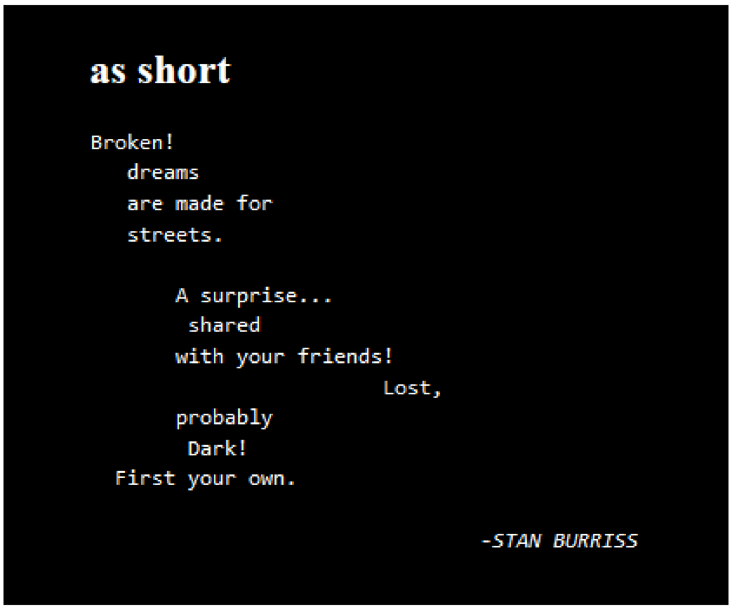

This is a sample paragraph which breaks here
and starts a new line here and can be breaks
anywhere according to the purpose

Figure 1. 13 Output of <br> tag

The HTML '**pre**' element is used to displays the white spaces, line breaks and every formatting you make inside the element. (*click here for live example*)

```
(e.g.)  <h2>as short</h2>
        <pre>
        Broken!
            dreams
            are made for
            streets.

                A surprise ...
                  shared
                with your friends!
                                    Lost,
                probably
                  Dark!
          First your own.
                                                  <em>-STAN BURRISS</em>
        <pre>
```


```
as short

Broken!
   dreams
   are made for
   streets.

       A surprise...
        shared
       with your friends!
                        Lost,
       probably
        Dark!
  First your own.

                              -STAN BURRISS
```

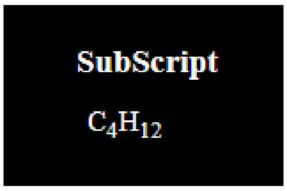
Figure 1. 14 Output of <pre> tag

2.8 Superscript, Subscript and Small Text

When you need to markup certain bits of content as having a different meaning than the others, you can use some HTML elements like ‘**sub**’, ‘**sup**’ and ’**small**’.

The HTML ‘**sub**’ element is used to markup subscripts, which are characters that are set below the baseline of text. (*click here for live example*)

(e.g.) <p>C<sub>4</sub>H<sub>12</sub></p>


SubScript

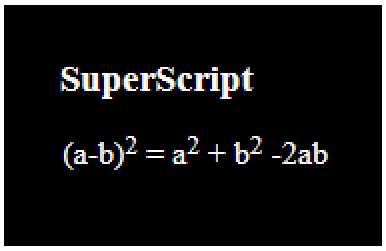
C_4H_{12}

The HTML ‘**sup**’ element is used to markup superscripts, which are characters that are set above the normal baseline of text.

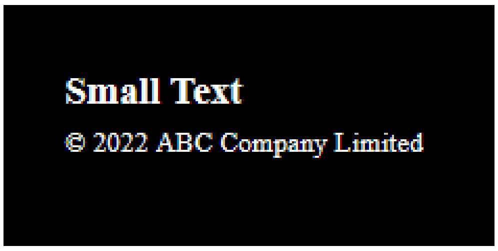
(e.g.) <p>(a-b)<sup>2</sup> = a<sup>2</sup> + b<sup>2</sup> -2ab</p>

The HTML ‘**small**’ element is used to markup smaller text, which is to convey that this text has a very little prominence, and it is often used to markup copyrights and side-comments.

(e.g.) <small>&copy; 2022 ABC Company Limited<small>


SuperScript

$(a-b)^2 = a^2 + b^2$ -2ab


Small Text

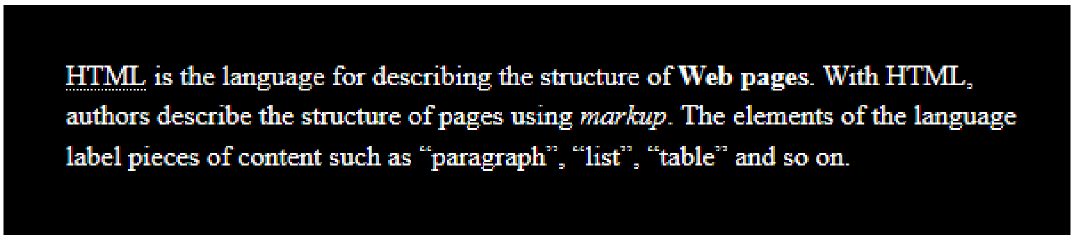
© 2022 ABC Company Limited

NOTE: For copyrights symbol (&copy;)HTML character entity is used

2.9 Inline and Block-level Elements

HTML elements were categorized as either “block-level” or “inline-level” elements.

Inline-level elements are those which only wraps around phrases of content, and it does not start a new line and only take up much width as necessary. (*click here for live example*) (e.g.) *<abbr title="Hyper Text Markup Language">*HTML*</abbr>* is the language for describing the structure of *<b>*Web pages*</b>*. With HTML, authors describe the structure of pages using *<i>*markup*</i>*. The elements of the language label pieces of content such as *<q>*paragraph*</q>*, *<q>*list*</q>*, *<q>*table*</q>* and so on.



HTML is the language for describing the structure of **Web pages**. With HTML, authors describe the structure of pages using *markup*. The elements of the language label pieces of content such as "paragraph", "list", "table" and so on.

Block-level elements occupies the entire horizontal space of its parent element and vertical space equal to the height of its contents, thereby creating a "block". It always starts on a new line and takes up the full width available. (*click here for live example*)

(e.g.) *<h2>*HTML - Hyper Text Markup Language*</h2>*

*<p>*HTML is the most widely used language on Web to develop web pages. Some of applications of HTML are*</p>*

<ul>

*<li>*Web pages development*</li>*

*<li>*Internet Navigation*</li>*

*<li>*Responsive UI*</li>*

*<li>*Offline support*</li>*

*<li>*Game development*</li>*

</ul>

<p>With HTML, authors describe the structure of pages using markup. The elements of the language label pieces of content such as "paragraph," "list," "table," and so on.</p>

See more (*List of Inline-level Elements*) and (*List of Block-level Elements*) on the MDN Web Docs.

3 Power and Features of HTML

3.1 Debugging HTML

When writing HTML code, what if an error occurs that means you've done something wrong, so your code doesn't work or either it is working but not quite how you wanted it to be. To find what’s wrong in the code Is done with the help of browser developer tools.

Usually, many modern browsers have developer tools, which includes inspect feature, and that helps to debug the code. You can see MDN Web Docs to know more about browser developer tools and how it works.

3.2 HTML Attributes

An HTML attribute is an associated property of an HTML element which is used to extend the capability of an HTML element. It is always specified before the lesser than symbol (**>**) of start tag. (*click here for live example*)

(e.g.) *<textarea cols=”40” rows=”10”></textarea>*

An element can have more than one attribute and each has its own value, and separated just by white spaces. Although some HTML attributes doesn’t have values and they are known as *Boolean Attributes.* If a Boolean attribute is present, its value is true and if it’s not, its value is false. (*List of Boolean Attributes*)

(e.g.)
```
<ol reversed>
        <li>Sugar</li>
        <li>Rice</li>
        <li>Wheat</li>
        <li>Salt</li>
</ol>
```

There are some HTML attributes which can be used with all the HTML elements and they are called the Global attributes. The most commonly used attribute is the ‘**class**’ attribute, it gives you a way to attach a reusable name to any element. So, you can address that name in CSS to apply styling to all the elements have that name. Each class name is separated by a white space like attributes, and you can use the same name for other elements. (*List of Global Attributes*)

(e.g.) *<p class=”para”>*This is a purple-coloured italicized line*</p>*

There is one attribute which is similar to the ‘**class**’ attribute which is ‘**id**’ attribute but id’s have their unique names, they are allowed to be used once in the entire HTML page.

(e.g.) *<p class=”para” id=”line2”>*This is a green-coloured italicized line*</p>*

These (ids) are often used to target specific elements to apply styling or do some graphics with JavaScript.

As you can saw before a global attribute named ‘**dir**’ is used to change the direction of text, or word, or sentence from right to left. That’s because it gives a way to tell the browser which direction the text flows inside a word or a sentence. It has three values ‘**ltr**’ is to be used for languages that are written from the left to the right, and ‘**rtl**’ is to be used for languages that are written from the right to the left, and ‘**auto**’ which lets the user agent decide.

3.3 Commenting in HTML and its Evolution

Another way programmers keep code easy to understand is by “documenting it”, by putting comments in the code to explain what’s going on. In HTML, anything comes between ‘**<!--**' and ‘**-->**’ is an **HTML comment**. The browser ignores comments as it renders the code. In other words, they are not visible on the page - just in the code. (*live example*)

(e.g.) *<p>*Comments written in HTML code is ignored by browser when rendering the code.*</p>* <!-- Comments can’t be visible in browser -->

As you might wonder, some of the HTML elements are very short like *<p>*, *<i>* and *<b>*. And others like *<blockquote>* and *<address>* are longer. That is because HTML was invented in the late 80s and in the early 90s, when computer scientists are obsessed with counting every single bit and bytes.

Along the way, as HTML evolved the web industry switched to believing that it’s much better to write all the elements with lowercase letters. The newer HTML elements always have both an opening and a closing tag.

3.4 Character Entity

Some characters are reserved in HTML, if you want to display a reserved character inside the content, you must use **Character Entity** otherwise the browser renders it as a piece of HTML code. (*see more on MDN web docs*)

These are some of the characters and their entity names:

Description	**Character**	**Entity Name**
less than	**<**	**&lt;**
greater than	**>**	**&gt;**
ampersand	**&**	**&amp;**
double quotation mark	**“**	**&quot;**
single quotation mark	**‘**	**&apos;**
copyright	**©**	**&copy;**

4 Linking and Navigation

In the 1980s, people who were inventing new ways for computers to help us do our work obsessed thinking and talking about the link. Obsessed with hypertext, hypermedia, hyperlink.

4.1 Links

To make link in HTML, ‘**a**’ element is used. On the opening tag, an attribute named ‘**href**’ creates a hyperlink to web pages, files, email addresses, locations in the same page, or any URLs.

a – stands for *anchor*

href – stands for *Hypertext Reference*

In between the opening and closing ‘**a**’ tags, put whatever it is that you want to be clickable. It can be text or image or any other HTML element. (*click here for live example*)

(e.g.) *<p><a href=”https://codepen.io/”>*CodePen*</a>* is a social development environment for front-end designers and developers.*</p>*

By default, the ‘**a**’ element is an inline element and can be wrapped around anything. It has some default styling when displaying in the browser. When the link is unvisited it appears to be underlined and blue in color, when the link is visited it appears to be underlined and violet in color, and when the link is active it appears to be in underlined and red in color.

The ‘**a**’ tag has another attribute named ‘**target**’ which specifies where to open the hyperlink. It has can have values from the following four: ‘**_self**’, ‘**_blank**’, ‘**_parent**’ and ‘**_top**’.

(e.g.) *<p>*To clarify your doubts with other Web developer *in <a href="https://stackoverflow.com/" target="_blank">*Stack Overflow*</a></p>*

4.2 URL Paths

URLs are classified into two absolute and relative URLs. **Absolute** URLs (full web address) are the one which links to something out on the web someplace or some other website, and it must include from the HTTP/HTTPS protocol.

(e.g.)
```
<h2>Absolute URLs</h2>
<p><a href="https://www.w3.org/">W3C</a></p>
```

Relative URLs are the ones used for linking to something that's on the same site, or in the same domain name as the page that has the link. (*click here for live examples*)

(e.g.)
```
<h2>Relative URLs</h2>
<p><a href=""/collection/yrbgjp">HTML Beginner Guide</a></p>
```

The '**/**' slashes in a URL means look inside there, or go one level deeper. To create a relative URL just leave off the domain name, and include the initial slash at the very beginning. That tells the browser, go to the root level of the file structure, the outermost top level and start from there.

4.3 Navigations

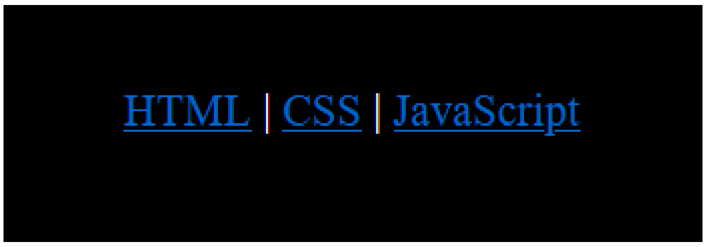
The HTML '**nav**' element is for marking up navigation. It tells the browser which sets of links are parts of the site navigation. It is only intended only for the major block of navigation links.

(e.g.)
```
<nav class="main-menu">
        <a href="https://developer.mozilla.org/en-US/docs/Learn/HTML"
        target="_blank">HTML</a> |
        <a href="https://developer.mozilla.org/en-US/docs/Learn/CSS"
        target="_blank">CSS</a> |
```

*<a href="https://developer.mozilla.org/en-US/docs/Learn/JavaScript" target="_blank">*JavaScript*</a>*

</nav>

Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content. (*click here for live example*)


HTML | CSS | JavaScript

One more set of links, something that can be found at the bottom of the page in the footer section, that kind of links that don't need to be in a <nav> element.

5 Images and Graphics

In the beginning, the web was just text, and it was really quite boring. Fortunately, it wasn't too long before the ability to embed images (and other more interesting types of content) inside web pages was added. Undoubtedly images can improve the design and the appearance of a web page.

5.1 Images

To put an image on a webpage we use HTML '**img**' element and it is often called as *image* element. It doesn't have a closing tag and it contains only attributes in the opening tag. There are two requires attributes to be included with the image element one is the '**src**' attribute, sometimes spoken as its full title, source and contains the path pointing to the image you want to embed in the page, which can be a relative or absolute URL.

The another required attribute is the ‘**alt**’ attribute which contains the alternative text for the image, which can be displayed where the image cannot be seen/displayed or takes a long time to render because of a slow internet connection. (*click here for live example*)

(e.g.) *<img src="https://images.unsplash.com/photo-1653274517623-81ad3c65b9f5?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHx0b3BpYy1mZWVkfDI1fEpwZzZLaWRsLUhrfHxlbnwwfHx8fA%3D%3D&auto=format&fit=crop&w=500&q=60" alt="A Cockatoo resting on a branch">*

There are two more attributes used with ‘**img**’ tag, they are ‘**width**’ and ‘**height**’ which specifies how the image is displayed in width and height. These attributes don’t include a unit with it, the values of attributes are just numbers and it is then understood as pixels by the browser, and it doesn’t care about the order of attributes.

(e.g.) *<img src="https://images.unsplash.com/photo-1653274517623-81ad3c65b9f5?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHx0b3BpYy1mZWVkfDI1fEpwZzZLaWRsLUhrfHxlbnwwfHx8fA%3D%3D&auto=format&fit=crop&w=500&q=60" alt="A Cockatoo resting on a branch" width=300 height=200>*

Including the *width* and *height* attribute on images will fix articles jumping around when image loads.

5.2 Supporting Image Formats

To put an image on a webpage, it has to be in a file format that web browser will understand and there are several options. The main goal is to have the highest quality possible image with smallest file size possible. Each file format takes a different approach, and each uses a different technique for compressing the image.

The most famous file formats commonly used on web today are , **GIF**, **SVG**, **JPG** and **PNG**.

The ‘**GIF**’ is one of the oldest and good at compressing illustrations that have large areas of a single color. It has a limited color space of 256 colors, so your image can only use these colors. Among these four, GIF is famous for making small movies, the animated GIF.

‘**SVG**’ (Scalable Vector Graphics) is great for logos, icons and other kind of illustration. They are made up of vector and math, these images can scale to be physically quite large or very small without even losing any quality. It won’t get pixelated because there aren’t any pixels and that makes the image file very small.

‘**JPG**’ is known for being a great format for compressing photographs. Most digital cameras save photos as JPG files. When we put a JPG image on the web, we want to make sure that either a human or robot resizes the image to an appropriate physical size and compresses it for the web. JPGs can be compressed to much smaller sizes, by simplifying the color information in the file, we can pick a balance between the quality and the file size.

‘**PNG**’ are newer format for the web than GIF or JPG . They’re a good solution when you have photograph and you need part of a photograph to be transparent.

5.3 Responsible and Responsive Images

Using the power of HTML, you can deliver different image files according to screen size or according to the viewport width or by the operating system and network speed. Sometimes, you can even give completely different images altogether.

5.3.1 Responsible Images

Basically, there are bunch of screens out there with a pixel density of 2x, 3x, 4x where more data can be displayed by the screen. The simplest way to support these different screens is to create multiple copies of your images in different resolutions and then tell the browser that these copies are available.

The browser will look at the screen density, the network connection, the user settings and decide which image to use. Even if someone has a high-resolution screen, the browser might decide to download a lower resolution image according to their network speed. (*click here for live example*)

(e.g.) *<img src= "https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-for-Beginners/main/Images/Responsive/cockatoo_480p.jpg"*
alt="A Cockatoo resting on a branch"
width="480"
srcset="https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-for-Beginners/main/Images/Responsive/cockatoo_960p.jpg 2x,
https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-for-Beginners/main/Images/Responsive/cockatoo_1440p.jpg 3x,
https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-for-Beginners/main/Images/Responsive/cockatoo_1920p.jpg 4x">

This is a great solution for providing different size images to handle retina and high DPI Screens. The '**srcset**' attribute contains a string containing a comma-separated list of one or more image candidate strings. Each image candidate string must begin with a valid URL referencing a non-interactive graphic resource. This is followed by whitespace and then a condition descriptor that indicates the circumstances in which the indicated image should be used.

In the above example, inside the srcset attribute you can see 2x, 3x, 4x after the absolute URLs. These are the one of the two types of condition descriptor known as **pixel density descriptor**.

5.3.2 Responsible Width

If you want to consider the width of the viewpoint and to choose a file based on the size of the screen as well as the density. This can also be done with the '**srcset**' attribute like in the above example. But, instead of using pixel density descriptor you should use another type of condition descriptor known as **width descriptor**. It can be done by adding the width in pixels followed by a lowercase ‘**w**’. (*click here for live example*)

(e.g.)
```
<img src= "https://raw.githubusercontent.com/mahendraKumara/HTML-
Guide-for-Beginners/main/Images/Responsive/cockatoo_static.jpg"
alt="A Cockatoo resting on a branch"
width="480"
srcset=" https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-
for-Beginners/main/Images/Responsive/cockatoo_480p.jpg 480w,
                https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-
for-Beginners/main/Images/Responsive/cockatoo_960p.jpg 960w,
               https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-
for-Beginners/main/Images/Responsive/cockatoo_1440p.jpg 1440w,
             https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-
for-Beginners/main/Images/Responsive/cockatoo_1920p.jpg 1920w">
```

Now the browser will dynamically choose which image to display based on a combination of device density and viewport width.

5.3.3 Responsive Pictures

Sometimes you want to use a different image on a small screen versus a big screen. Maybe we want the image to be tall and narrow on mobile where it’s short and wide on desktop, or even use a completely different photo altogether.

The image element alone, even with the ’**srcset**’ or ‘**sizes**’ attribute can do that, so ‘**picture**´ element is used. It gives full control over images the inner ‘**source**’ element which haves the control over multiple image files by using ‘**media**’ and ‘**srcset**’ as attributes.

The ‘*media’* attribute has the condition of the screen width in pixels, when the condition becomes true it picks the one of the image files from ‘*srcset’* attribute and display on the screen. If the condition is false then another **source** elements takes place and if all the source condition fails the ‘**img**’ element shows the image file in it or it’s alternative text. (*click here for live example*)

(e.g.)

```
<picture>
    <source media="(min-width:600px)"
    srcset="https://raw.githubusercontent.com/mahendraKumara
    /HTML-Guide-for-
    Beginners/main/Images/Responsive/cockatoo_720p.jpg">
    <source
    srcset="https://raw.githubusercontent.com/mahendraKumara
    /HTML-Guide-for-
    Beginners/main/Images/Responsive/cockatoo_320p.jpg">
    <img
    src="https://raw.githubusercontent.com/mahendraKumara/H
    TML-Guide-for-
    Beginners/main/Images/Responsive/cockatoo_static.jpg"
    alt="A Cockatoo resting on a branch" width="480">
</picture>
```

Normally, a person looking at a web page through one screen. Their viewport is staying in one size and they just get the one image file, which is the one that's best for them. And none of the rest of the files are downloaded, which is the whole points to save downloads.

HTML gives you a powerful set of options for optimizing images on the web. So, we can send the smallest file possible while still publishing gorgeous images, that are sometimes displayed quite large.

5.4 Figure and Figcaption

There are two more elements related to images, or really rather, related to figures. The '**figure**' and '**figcaption**' elements are handy for anything that appears as a figure illustrating something, or anything that's a demonstration of a concept which needs a caption. (*click here for live example*) (e.g.) *<figure>*

<img src=”https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-for-Beginners/main/Images/Responsive/racoon.jpg" alt="racoon looking through a fence" width="480">

*<figcaption>*A Cute little racoon looking through a wooden fence*</figcaption>*

</figure>

6 Media in HTML

In HTML multimedia are known as media, and it comes in many different formats. It can be almost anything you can hear or see, like images, music, sound, videos, records, films, animations and more. They are stored in media files and web pages often contain multimedia elements of different types and formats.

6.1 Audio

In HTML ‘**audio**’ element is used to put audio files on a webpage. The HTML is fairly simple and it can trigger the browser to create an entire audio play or interface for us. It is a newer element so it has both opening and closing tag. Then just like the *image* element it has a ‘**src**’ attribute to provide the URL to the audio file.

<audio src=””> </audio>

To make the audio element play an audio, you need to use the ‘**control**’ attribute to provide some controls. Using the controls that are built into the browser is optional, or you could make your own custom controls using JavaScript and the HTML media element API.

<audio src=”” controls> </audio>

There are several other attributes that can be used with *audio*. They are ‘**loop**’ which will make the audio loop, and ‘**autoplay**’ which will play the audio automatically as soon as the page loads.

<audio src=”” controls autoplay> </audio>

<audio src=”” controls> </audio>

The purpose of having both opening and closing tags, you can add more than one audio file format by using the ‘**source**’ element like in the *picture* element.

(e.g.)
```
<audio controls>
    <source
    src="https://raw.githubusercontent.com/mahendraKumara/HTML-
    Guide-for-Beginners/main/Audio/sampleMP3.mp3"
    type="audio/MPEG">
    <source
    src="https://raw.githubusercontent.com/mahendraKumara/HTML-
    Guide-for-Beginners/main/Audio/sampleOGG.ogg"
    type="audio/OGG">
</audio>
```

By using separate source element, you can make multiple files. Then the browser will use the first file that it supports from the list of sources. Between end the *audio* element with closing tag you can provide some fallback text, that will only be displayed if the audio element is not understand by the browser at all. (*click here for live example*)

6.2 Video

In HTML, one way to put video content on a webpage is to use the ‘**video**’ element in HTML. Its syntax and attributes are similar to the *audio* element. There are bunch of different codecs which a video file might be encoded, from those MP4, OGV, WEBM are formats supported by HTML. (*click here for live example*) (e.g.) *<video controls>*

<source src="https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-for-Beginners/main/Audio/sampleMP3.mp3" type="audio/MPEG">

<source src="https://raw.githubusercontent.com/mahendraKumara/HTML-Guide-for-Beginners/main/Audio/sampleOGG.ogg" type="audio/OGG">

</video>

Video files contain a lot of data and can get incredibly big when they aren't compressed, far too big to big to travel across the internet. So, any internet video is using a pretty hefty mechanism for smashing all the data int the smallest possible package.

6.3 Captions and Subtitles

On the web you have the power that you need to provide content in multiple ways simultaneously. You can also add a caption to video by using the '**track**' element and point it to a text file. The video player in the browser will provide all the functionality, so a viewer can turn captioning on and off or switch between set of subtitles.

Like video and audio, there are many formats for captions. But on the web, you want to use '*webvtt'* (Web Video Text Track) and its basically a plain text file with '*vtt'* extension that uses a certain convention for providing the information.

To get captions to show up on your video, put the **track** element inside the '**video**' element. Just like the *source* element, its part of the list of options for browser to use while rendering the video player. (*click here for live example*)

On the '**track**' element, the subtitle file is directly pointed inside the '**src**' attribute and other than *src* attribute it has '**kind**', '**label**', '**srclang**' attributes.

(e.g.)

```
<video controls>
        <source
        src="https://raw.githubusercontent.com/mahendraKumara/HTML-
        Guide-for-Beginners/main/Audio/sampleMP3.mp3"
        type="audio/MPEG">
        <track label="English" kind="subtitles" srclang="en"
        src="https://raw.githubusercontent.com/mahendraKumara/HTML-
        Guide-for-Beginners/main/Subtitles/ElephantsDream.vtt" default>
</video>
```

Using the '**kind**' attribute to tell the browser this is a caption, then the '**label**' attribute to show up in the player as a label for the choice and '**srclang**' attribute to specify which language and '**default**' attribute specify that this track Is the one to use by default when the user turns on the captions. (*click to see more values of kind attribute*)

6.4 Embedding Media through <iframes>

Embedding is taking content from one site and placing it into the middle of a page on another site. There's all kind of content that you might want to embed onto a page. And it's common to embed something complex from a service that takes care of all the hard parts. (*click here for live example*)

(e.g.)

```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/7b3Q5WiYtvg" title="YouTube video
player" frameborder="0" allow="accelerometer; autoplay; clipboard-write;
encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>
```

The HTML '**iframe**' element has several attributes on it including height and width. So, you could adjust those but remember *iframe*s are powerful. (*List of Attributes in iframe*)

7 Identifying HTML Content and its Structure

There are more ways to identify content in HTML, you can do that by the structure of HTML document and the language used in it or the language it supports. HTML files can be rendered even if the structure changes but it could not be the output you expected or want to be.

7.1 Supporting Languages

There are several tools in HTML you want to use to make sure it's clear which language your content is in. If the things are set up properly, search engines will know when to list your sites in which language. The right dictionaries will show up for the spell checkers, when the content is read aloud by a browser, it's more likely that the words will get pronounced properly.

A global attribute named '**lang**' is what you use to do this, if the whole page is in the same language, then it's pretty simple to set the *lang* attribute on the outermost element that wraps all the other elements, which is the '**html**' element.

There are many options for the value of the *lang* attribute, not just to convey the language or the region-specific version of the language, and to convey which kind of alphabet is being used to write a language.

If the webpage has more than one language on it, you want to specify the language for each particular part of the content. Most languages in the world flow from a left to right direction in a horizontal direction, but some flow to right to left. So, using '**dir**' attribute which direction is your content flows is specified.

```
<html lang="en-US" dir="ltr">
<html lang="ar" dir="rtl">
```

Well, every human language is written in a script, the alphabet or the set of characters. The original digital character sets were pretty limited and only supported the Latin alphabet.

ASCII, created in the 1960s, it only has 128 characters, basically the characters from an English language typewriter. These days, most projects use Unicode, specifically UTF-8, which basically means all our content is encoded according to this giant UTF-8 specification.

To specify the charset for HTML, simple write meta charset equals UTF-8. Then include that ‘**meta**’ element on every page in your site, inside the head element.

```
<head>
        <meta charset=”UTF-8”>
</head>
```

7.2 Generic Elements <div> and <span>

Sometimes you just need a way to wrap a group of elements or to mark a phrase. Sometimes there is no meaning to it, you just need a way to target part of the DOM (Document Object Model) with CSS or JavaScript. Well, there are two elements for that, and it can be used anytime, they are ‘**div**’ and ‘**span**’.

‘**div**’ is a block-level element and ‘**span**’ is an inline element. They both do the same thing, which is nothing. Until some CSS or JavaScript is pointed at them. (*click here for live example*)

(e.g.)
```
<div>
        <p><b>‘div’</b> is a <i>block-level</i> element and <b>‘span’</b> is
        an <i>inline-level</i> element. They both do the same thing, which is
        nothing. Until some CSS or JavaScript is pointed at them. <span>Both
        are known as generic elements.</span></p>
</div>
```

7.3 The Structure of HTML Page

HTML files are key building block of the web. Anytime anyone wants to go to a website, they start by opening up some kind of web browser or web view and going to the URL. Visual styling is in CSS files, JavaScript is in separate files, plus the images, video, audio files and advertisements. And what a user actually sees once a page is loaded, these days it's actually assembled from a mishmash of different files.

No, matter how the website is engineered, the web itself works this same every time. If a user goes to a URL, hitting the URL send a request for an HTML file, a server returns a single HTML file, then the browser reads the HTML file and does what is says.

HTML file, is the first file that's returned in response to a request for a webpage, that HTML file is pretty important. It functions as the headquarters for everything else that happens after the first moment of the site loading. Once the HTML file is built, there are several key parts that every web page must have. First, the file must start with a '**doctype**'. This is a little statement declares which era the HTML file is from.

```
<!doctype html>
```

Then everything else on the page is wrapped within an '**html**' element. By putting the opening *html* tag at the top and a closing *html* tag all the way down to the bottom.

```
<!doctype html>
<html>
    ….
</html>
```

Next, if the whole page is in the same language, then you want to declare the language using *lang* attribute and use dir attribute to tell which direction the content flows.

```
<!doctype html>
<html lang="en-US" dir="ltr">
 ….
</html>
```

Inside the ‘**html**’ element, two major parts of elements where everything goes, which is the ‘**head**’ and the ‘**body**’. The *head* element is for all the metadata, and all the stuff about the document that a browser needs to know but it will not be displayed on the page. The *body* element is for all the content which is to be displayed to the viewer, each content is wrapped into many different elements. (*click here for live example*)

```
<!doctype html>
<html lang="en-US" dir="ltr">
        <head>
                <meta charset=”UTF-8”>
        </head>
        <body>
                <h1>Structure of HTML Document</h1>
                <p>The HTML file must start with a <b>doctype</b>. This is a little
                statement declares which era the HTML file is from.</p>
                <p>Everything else on the page is wrapped within an <b>’html’</b>
                element. By putting the opening html tag at the top and a closing html
                tag all the way down to the bottom.</p>
                <p> Inside the <b>‘html’</b> element, two major parts of elements
                where everything goes, which is the <b>‘head’</b> and the
                <b>‘body’</b>.</p>
                <ul>
                        <li>The head element is for all the metadata, and all the stuff
                about the document that a browser needs to know but it will not be
                displayed on the page.</li>
                        <li>The body element is for all the content which is to be
                displayed to the viewer, each content is wrapped into many different
                elements.</li>
                </ul>
        </body>
</html>
```

7.4 Document Head

The ‘**head**’ element contains the information about the website that you want a browser to know, for this ‘**meta**’ elements are used inside the *head* element. It conveys the metadata about the pages.

Another required element is the ‘**title**’ element every page should have, this is not the title that’s gets displayed as content. This is a title for the webpage that’s used by the browser.

```
<head>
    <title>Title of a Web Page</title>
</head>
```

The *title* element shows up as the name on a browser tab, or as a bookmark title when someone bookmarks it. The name of the page, if it’s listed under top sites of the browser references this page, it was the title that’s defined in the ‘**title**’ element.

The ‘**meta**’ element can be used to define all kinds of settings, and one common use is to tell the browser the layout has been morphed to fit small screens, that it is a responsive website.

```
<head>
    <meta name=”viewport” content=”width=device-width, initial-scale=1”>
</head>
```

Without this *meta* tag, the browser assumes the page is using an older technique for layout, that it’s a desktop layout that needs to be shrunk down to fit on a phone. You probably also want to add a description of the site. The description will show up in search engine results.

```
<head>
    <meta name=”description” content=”A description of the site shown in
    search engine results”>
</head>
```

Another important element is ‘**link**’ element, it is used to link to a range of other assets that you want to load, like CSS files, fonts and favicons. The ‘**rel**’ attribute of this element tells the browser which kind of asset it is. And the ‘**href**’ attribute is used to provide the URL.

```
<head>
        <link rel=”stylesheet” href=”style.css”>
        <link rel=”icon” href=”favicon.ico”>
</head>
```

By using *link* element, you can also link to preload a font file too. The browser will always load the files in the order in which they’re listed in the HTML document.

```
<head>
        <link rel=”preload” href=”myFont.woff2” as=”font” type=”font/woff2”
        crossorigin=”anonymous”>
</head>
```

The HTML *head* is the place to get everything connected and set up to make sure other assets are loaded and to provide data about the page to other sites and platforms.

NOTE: HTML ‘head’ always comes in the top next to the opening *html* tag and there should be only one *head* element to be present for the entire html document.

7.5 Structural Elements of HTML

Webpages can and will look pretty different from one another, but they all tend to share similar standard components, unless the page is displaying a full screen video or game, is part of some kind of art project, or is just badly structured.

Websites often display content in multiple columns (like a magazine or a newspaper). To implement such semantic mark up, HTML provide six major elements:

- main

- header
- footer
- article
- section
- aside

The '**main**' element wraps around the main content of the page. It's only used once per webpage, and it tells the browser, this is where the main content is.

The '**header**' and '**footer**' element are used to mark places on the page where the content is a header or a footer. Most webpages have a header at the top. Maybe with the logo, the site name, the navigation bar or icons.

Many webpages end with a footer at the bottom, with list of links, some have copyrights information, or extra information about the company, that would naturally be wrapped in a '**footer**' element.

```
<footer>
        <a href=”#”>Privacy Policy</a>
        <a href=”#”>Terms of service</a>
</footer>
```

An '**article**' element is wrapped around any instance of an article. It might be a long-written article or blog post, or it might be a very short snip.

The '**section**' element is used to wrap around sections of content. If you have a long essay for instance, that's broken up into chunks with sub headlines, there you can use the section element to mark each segment.

The '**aside**' is for marking something that's off to that side. Perhaps in a side bar, or any content that's kind of not in the main attraction. It could be an insert panel that goes with a big article giving additional information, but isn't quite part of the flow of the article itself. These elements are used together, nested inside each other in patterns and combinations that gives the content of a webpage its true structure.

8 Forms and Interactive Elements

Forms on the web are how we all get things done. Using forms, a person can log into sites, buying things, request a search, add content to a site. An HTML form is used to collect user input. The user input is most often sent to a server for processing.

8.1 Basics of HTML Form

The ‘**form**’ element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc. It has both the opening and closing tag. It has plenty of attributes in it, but some common attributes are ‘**method**’ and ‘**action**’

The ‘**action**´ attribute contains the URL that to be processed after form submission. The ‘**method**’ attribute decides which HTTP method the form is submitted according to the value in it. (*Full List of Attributes for Form*)

The HTML *form* element can contain one or more of the following elements: <button>, <datalist>, <fieldset>, <input>, <label>, <legend>, <meter>, <optgroup>, <option>, <output>, <progress>, <select>, <textarea>.

One of the most used *form* elements is the ‘**input**’ element and it doesn’t have a closing tag. The *input* element can be displayed in several ways, depending on the value of ‘**type**’ attribute used in it. Some common values of *type* attribute are: ‘**text**’, ‘**button**’, ‘**email**’, ‘**password**’, ‘**number**’ and ‘**submit**’. (*click here for live example*)

```
(e.g.)  <form method="GET" action="">
                <input type="text" placeholder="username">
                 <input type="button" value="Continue">
        </form>
```

In the above example, as you see there are attributes other than *type*. The ‘**placeholder**’ attribute allows you to show pre-populated text through its value. The ‘**value**’ attribute in the button type input helps you change the display text over a button. (*Full List of Attributes for input*)

8.2 Additional Elements in Form

The ‘**textarea**’ element is used to collect a paragraph or whole article of text. And it’s similar to the *input* element. You can also put a limit the size of this element by ‘**cols**’ and ‘**rows**’ attribute. (*click here for live example*)

```
<form method="GET" action="">
        <textarea name=”comment” cols=”30” rows=”15”>
         </textarea>
</form>
```

9 Structuring Tabular Data

Generally, tables organize text into rows and columns, which can make the text easy to type, edit, and format while spacing it correctly in your document. Tables organize text into cells, where a cell is the intersection of a row and a column.

9.1 When to use Tables

While comparing prices of things that are for sale, population data by town, election results, product comparisons, schedules, bit of information collected that you want people to be able to quickly compare, this is what tables are used for.

The data might be numbers, text, images or anything. There’s a semantic reason for data to be organized into a table, that meaning is added by an arrangement into rows or columns, the table conveys a relationship between the data cells and the header cells between one column or row and the next.

9.2 Building HTML Tables

To create a HTML table, several different HTML elements used in just the right combination.

- table
- tr
- th
- td

The ‘**table**’ element defines an HTML table, which allow web developers to arrange data into rows and columns. It has both an opening and closing tag which wraps around all the markup used to create a table.

The ‘**tr**’ element stands for table row. It wraps around a set of elements, defining them as belonging to the same row. The ‘**th**’ element stands for table header and it defines a header for a column. The ‘**td**’ element stands for table data, it marks up the cells of data. (*click here for live example*)

(e.g.)
```
<table class="dogs-species">
        <tr>
                <th>DOG</th>
                <th>WEIGHT (in pounds)</th>
        </tr>
        <tr>
                <td>Bernese Mountain Dog</td>
                <td>70 - 115</td>
        </tr>
        <tr>
                <td>Chinook</td>
                <td>158 - 90</td>
        </tr>
    </table>
```

You can also create a complex HTML table, ways to have our content span multiple rows, or multiple columns. You can define a header, body and footer for the table or add a caption like a heading or a text in *figcaption* element.

___ ____ ____ ___

References

- MDN Web Docs
- W3C

Resources

- CodePen - Online Code Editor
- GitHub
- Stack Overflow
- FileSamples
- Unsplash - Free Background Images