Week 1- day 1

**HTML5 Introduction**

**Done**

**Table Of Contents**

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#What_you_will_learn)
  + [Useful resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#Useful_resources)
    - [What is HTML?](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#What_is_HTML?)
    - [HTML Versions](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#HTML_Versions)
    - [Choosing Your HTML Editor](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#Choosing_Your_HTML_Editor)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1139#Feedback)

Last updated : March, 31st 2021

**What You Will Learn**

* The Basics of HTML

**Useful Resources**

* [Difference between HTML and HTML5](https://www.geeksforgeeks.org/difference-between-html-and-html5/).
* [HTML5 | Semantics](https://www.geeksforgeeks.org/html5-semantics/).

**What Is HTML?**

HTML, an acronym for **HyperText Markup Language**, is a computer language for creating websites and web applications. Consisting mainly of series of codes usually written in a text file and saved as .html. Code written in the HTML language translates into a beautiful, well-formatted text or a combination of text and media when viewed through a browser.

British physicist Tim Berners-Lee first developed HTML in 1990. It had gone through so many evolutions since then that the most recent version can achieve far more than was imagined possible when Tim first invented the language.

**HTML Versions**

First, a quick rundown of all the HTML versions since HTML was invented.

* HTML 1.0: This was the barebones version of HTML and the first release of the language.
* HTML 2.0: This version was introduced in 1995. It gradually evolved, allowing extra capabilities, including form-based file upload, tables, client-side image maps, and internationalization.
* HTML 3.2: In an attempt to ensure the development of standards for the World Wide Web, in 1994 Tim Berners-Lee founded the World Wide Web Consortium (W3C). By 1997, they published HTML 3.2.
* HTML 4.0: Later in 1997, the W3C released HTML 4.0 — a version that adopted many browser-specific element types and attributes.
* HTML 4.0 was later reissued with minor edits in 1998.
* HTML 4.01: In December 1999, HTML 4.01 was released.
* XHTML: The specifications were introduced in 2000, and it was recommended to be used as the joint-standard with HTML 4.01. It incorporated XML to ensure code is properly written and to ensure interoperability between programming languages.
* HTML5: The W3C published HTML5 as a recommendation in October 2014 and later released HTML 5.1 in November 2016.

**Choosing Your HTML Editor**

If you are thinking of creating web pages in HTML, you will need an HTML editor. There are several benefits to using an HTML editor.

A good HTML editor will keep your code clean and organized. It will also detect when you open a new tag and automatically close it to avoid you having a buggy code and reduce how much typing you have to do. Today, most HTML editors allow you to preview your web page to see how it will look in a web browser using their WYSIWYG feature.

There are many free and paid HTML editors; below are some of the top options you can choose from:

* Sublime Text
* HTML-Kit
* CoffeeCup
* KompoZer
* Komodo Edit
* Notepad++
* Bluefish
* CodeLobster

## HTML5

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#What_you_will_learn)
  + [Useful resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Useful_resources)
    - [HTML files structure](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#HTML_files_structure)
    - [Basic Building Blocks of HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Basic_Building_Blocks_of_HTML)
      * [Introduction to Tags](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Introduction_to_Tags)
      * [Introduction to Elements](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Introduction_to_Elements)
      * [Introduction to Attributes](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Introduction_to_Attributes)
      * [Creating Headings in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Creating_Headings_in_HTML)
* [h1 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h1_heading)
  + [h2 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h2_heading)
    - [h3 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h3_heading)
      * [h4 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h4_heading)
        + [h5 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h5_heading)

[h6 heading](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#h6_heading)

* + - * [Creating paragraphs](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Creating_paragraphs)
      * [Formatting Text in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Formatting_Text_in_HTML)
      * [Ordered and Unordered Lists](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Ordered_and_Unordered_Lists)
      * [Nested lists](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Nested_lists)
      * [Hyperlinking](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Hyperlinking)
      * [Using Images](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Using_Images)
      * [Creating Tables](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Creating_Tables)
      * [Quotations in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Quotations_in_HTML)
      * [Using Comments in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Using_Comments_in_HTML)
      * [Using iFrames in HTML](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Using_iFrames_in_HTML)
      * [HTML Forms](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#HTML_Forms)
      * [Defining a form](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Defining_a_form)
        + [The input element](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#The_input_element)
        + [The label element](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#The_label_element)
    - [Conclusion](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Conclusion)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/24#Feedback)

Last updated : March, 31st 2021

## What You Will Learn

* The Basics of HTML
* Create your first Web page using HTML
* Understand the logic behind a webpage
* HTML tags and attributes

## Useful Resources

* [HTML Periodic Table of Tags](https://websitesetup.org/html5-periodical-table/).
* [W3Schools/Tags – List of all HTML tags ordered alphabetically](https://www.w3schools.com/tags/).
* [Mozilla/HTML Attributed – List of all HTML attributes ordered alphabetically](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes).
* [HTML Cheat Sheet](https://websitesetup.org/html5-cheat-sheet/).
* [Another HTML Cheat Sheet](https://htmlcheatsheet.com/).
* [HTML Color codes/Color Names – List of 140 HTML color names including HEX code, RGB value and HSL value](https://htmlcolorcodes.com/color-names/).

### HTML Files Structure

A HTML file is a text file with the .html extension.  
The code in the file needs to start with <!DOCTYPE html>.

An HTML element is defined by a start tag and an end tag, with the content inserted between:  
<element-type>My element value</element-type>

Notice how a tag is closed with the **/**  
Some tags don’t need to be closed, they’re called **orphan tags**

* All the HTML code is inside an HTML element (<html>).
* The HTML code is separated into two big parts, the <head> and the <body>.  
  The body is the visible part of the HTML code, while the head is here to add some config to the file.

Before content is added, most HTML files basically look like this:

<!DOCTYPE html>

<html>

<head>

<title>Title of the document</title>

</head>

<body>

The content of the document......

</body>

</html>

* Every HTML file begins with the opening tag <html> and ends with the closing tag </html>.
* The HTML file’s first line should declare the type of document so that the browser knows what HTML version you use. You see that before the HTML code begins, it starts with <!DOCTYPE html> .
* The section in the <head> tag usually contains information about the document such as its title, meta tags, and where to locate its CSS file – content that is not directly visible on the browser page.
  + Title - Example : <title>This is our page title</title>
  + This is our page title will be displayed in the browser tab. It’s also indexed as the title for the page when the search engine bots crawl your website.
  + Meta element: often used to specify the information, search engines can describe the content in their listings. This includes the description, keywords, author information, etc. The <meta> element also specifies the character set the HTML document uses.
* The section between the tag <body></body> is where the HTML file’s main content is located. It’s what the viewer will see in the browser. This includes everything from the first paragraph to hyperlinks, formatting, multimedia, and everything else.

### Basic Building Blocks Of HTML

When coding in HTML, it is essential to understand these building blocks. They include tags, attributes, and elements.

#### Introduction To Tags

In essence, **tags** separate normal text from HTML code.  
Tags are practically the building block of HTML – you can’t do HTML without tags! If you’re stuck on which tag to use, be sure to check out the Useful Resources above.

##### Opening And Closing Tags

Almost every open tag must be closed. Keep in mind that there are exceptions.

An example of a tag that does not have to be closed is an empty tag, such as the line break: <br>.

Tags are contained in less than (“<”) and greater than (“>”) angle bracket. Closing tags have a trailing slash that before the name of the tag being closed.

Example of an open tag: <b>. Example of a closed tag </b>.

When it comes to HTML, tags make the difference between whether your document is displayed as ordinary text or ‘transformed text’ . This ‘transformed text’ is a code that can display a series of things (hyperlinks, images, media, or other methods of formatting).

##### Bold & Italic Text

Let’s take a look at the word “He is a boy” as an example: \* In ordinary text format you get: He is a boy. \* In bold text format you get: **He is a boy**

To achieve what we have in the bold format, you have to use the <b></b> tag : <b>He is a boy</b>.

This sentence could also come out italicized.  
This is achieved using the <i></i> tag : <i>He is a boy</i>.

#### Introduction To Elements

In HTML, an “element” consists of the opening and closing tag and the content between the tags.

Example : <b>He is a boy</b>.  
The text “He is a boy” is surrounded by an open and closed tag. Everything, including the opening tag, the content, and the close tag is an **element**.

An element could be in a basic form or a complex form. Why? Because anything in between an opening tag and a closing tag is an element. It means that we can have elements within an element.

In our current example, “He is a boy” (<b>He is a boy</b>) is an element.

#### Introduction To Attributes

While HTML documents use tags for everything, we sometimes want to communicate additional information inside an element. In this case, we use an **attribute**. The attribute is used to define an element’s characteristics and it’s used inside the opening tag of the element. Attributes are made up of a name and a value.

Note that the value of an attribute is placed inside a quotation mark using the format

<tag attribute="value">Your Text</tag>

#### Example

<p align="center">He is a boy</p>  
In this example, we instruct that “He is a boy” is aligned in the center of the document.

#### Creating Headings In HTML

Headings play a significant role in the success of a website. Firstly, they make it easy for readers to scan the content of your pages. Secondly, and perhaps more importantly, they communicate your web pages’ structure to search engines and, therefore, often impact how your content is ranked in search engine results.

That said, it is essential to avoid using heading tags to make your text big or bold. Other tags can be used to make your text big (which we’ll get to later in this section). Instead, heading tags should be used solely for structure.

There are six heading tags in HTML: <h1> to <h6>, with the <h1> tag indicating the most important heading and the <h6> tag indicating the least important heading.

* HTML Headings Example:

Headings are used for titles and big texts

# H1 Heading

## H2 Heading

### H3 Heading

#### H4 Heading

##### H5 Heading

###### H6 Heading

#### Creating Paragraphs

Paragraphs can be created with the <p> tag.

#### Example

<p>This is your first paragraph.</p>

<p>This is your second paragraph, and you will be creating many more paragraphs.</p>

Keep in mind that writing in HTML is very different from writing in pure text. Therefore, if you break up text inside HTML without starting a new paragraph, it won’t matter when the browser displays the text. Instead, you want to use a line break, represented by the <br> tag.

#### Example

<p>This is a new paragraph.

And I want to use a number of new lines.

So I’m breaking it up.</p>

This **will not** come out as the following:

This is a new paragraph.  
And I want to use a number of new lines.  
So I’m breaking it up.

Instead, it will come out as this:

This is a new paragraph. And I want to use a number of new lines. So I’m breaking it up.

So, how do you break texts into new lines so that it would show like this:

This is a new paragraph.  
And I want to use some new lines.  
So I’m breaking it up.

By using line breaks.

<p>This is a new paragraph.<br>And I want to use several new lines.<br>So I’m breaking it up.</p>

If you want to create a horizontal line, you can use the <hr> tag.

<p>This is a new paragraph.<hr>And I want to create a horizontal line</p>

It will display like this:

This is a new paragraph.

And I want to create a horizontal line

#### Formatting Text In HTML

The next step is to format your text in HTML. This is where you can indicate whether you want your text to come out bold, italicized, underlined, subscripted, superscripted, etc. This is a basic guide, so this section won’t be the be-all-end-all for formatting. Instead, we will only include some basic formatting tags. The process for using other forms of formatting is simple – just find the tag you want and go ahead:

Using bold: <b>He is a boy</b> comes out as **He is a boy**

Using italics: <i>He is a boy</i> comes out as *He is a boy*

Underlining text: <u>He is a boy</u> comes out as He is a boy.

It’s worth noting that the <u> tag was deprecated in HTML 4.01 and was redefined to represent stylistically different text in HTML5. As a result, it is recommended to use CSS to indicate text that should be underlined. Since this article is a basic guide, we’re keeping it simple.

Using subscript: <sub>He is a boy</sub> comes out as He is a boy

Using superscript: <sup>He is a boy</sup> comes out as He is a boy

For other tags that can be used to format, take a look at **Useful Resources** at the beginning of the page.

##### Exercise 1

Create a structured HTML file as explained above.

Write about yourselves : your name, age, birthdate and why you decided to learn to code. Style this small text with paragraph, headings, italic and bold characters.

#### Ordered And Unordered Lists

Sooner or later, you will have to create lists. Lists could be ordered (i.e., numbered) or unordered (i.e., unnumbered).

Here is an example of an ordered list:

1. Item 1
2. Item 2
3. Item 3

Here is how to create it:

<ol>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ol>

Here is an example of an unordered list:

* Item 1
* Item 2
* Item 3

Here is how to create it:

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

If it is not already obvious. Here’s a breakdown:

The <li> tag is used to indicate each item on the list.  
When making a list, you can use the <ol> tag to indicate an ordered list (“o” = ordered and “l” = list) or the <ul> tag to indicate an unordered list (“u” = unordered and “l” = list).

#### Nested Lists

We can also have nested lists or a list within a list.

#### Example

* Item 1
  + Item 1 nested
  + Item 2 nested
  + Item 3 nested
* Item 2
* Item 3

This can be created with:

<ul>

<li>Item 1

<ul>

<li>Item 1 nested</li>

<li>Item 2 nested</li>

<li>Item 3 nested</li>

</ul>

</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

As you can see, you open another listing tag – ordered (<ol>) or unordered (<ul>) – depending on what you want before closing the item you want to be nested.

#### Hyperlinking

The web is one massively interconnected network of pages. If you create a website – whether internally or externally, or both – you will have to link to other pages. A link to an internal page on your website or an external page on the web is called a hyperlink. While people usually link a text, any HTML element – an image, for example – can be linked.

The <a> tag is used to define links in HTML, while the “href” attribute is used to specify a link’s destination.  
The link is then put into a quote after the “equal to” sign before the tag is closed.

#### Example

To create a hyperlink,

this:

<a href="https://websitesetup.org">YOUR LINK TEXT HERE</a>

will come out as: [YOUR LINK TEXT HERE](https://websitesetup.org/).

Now, assume you are linking to a local file in which you have all your HTML pages in the same folder. In this case, you don’t have to include a website URL. Just add the file path.

For example, if linking to a file titled about-page.html, your hyperlink becomes

<a href="about-page.html">YOUR LINK TEXT HERE</a>

which links to the about-page.html file.

If the HTML file you want to link to is local but different from the main folder, you specify the file path.  
For example, if the file is in the “files” folder under the main document where your current document is, your hyperlink goes something like this

<a href="files/about-page.html">YOUR LINK TEXT HERE</a>

You might want to specify how you want the link to be opened e.g., in a new window/tab. You will use the target attribute for that.

#### Example

To specify that the link should be opened in a new tab.

<a href="about-page.html" target="\_blank">YOUR LINK TEXT HERE</a>

To specify that the link should be opened in the same tab.

<a href="about-page.html" target="\_self">YOUR LINK TEXT HERE</a>

Other attributes can be used to communicate different ways a link should be opened:

* \_blank – Open in a new tab.
* \_self – Open in the same tab.
* \_parent – Open in the parent frame
* \_top – Open in the full body of the window
* framename – Open in a named frame

You can also have an image point to a link.

#### Example

<a href="https://websitesetup.org">

<img src="heisaboy.jpg" alt="He is a boy">

</a>

This tells the browser to display an image pulled from the image file “heisaboy.jpg” and have it linked to https://websitesetup.org.

The “alt” attribute represents what should be shown if the image cannot be displayed (due to browser settings or other stuff that prevents images from showing).

#### Using Images

Images are added to an HTML document with the <img> tag. The <img> tag is empty, so it doesn’t need to be closed.

#### Example

<img src="heisaboy.jpg" alt="He is a boy">

This is a basic example of telling the browser to display an image titled “heisaboy.jpg” pulled from the same directory as the HTML document.

The “alt” attribute tells the browser to show a text (known as an “alternative text”) called “He is a boy” if, for some reason the browser or internet settings prevents the browser from displaying images. If you want to pull an image in another directory or on an external site, the full address/path needs to be specified.

Going beyond the basics, we could also use other attributes to customize the image better. For example, the “style” attribute can be used to specify the width, the height, or both.

#### Example

<img src="IMAGE PATH" alt="YOUR ALTERNATIVE TEXT" style="width:Xpx;height:Ypx;">

(the values “Xpx” and “Ypx” should be replaced with the actual values in “px” (e.g. “20px”) you want for your image.)

#### Exercise 2

Work on the HTML file you previoulsy created

Make ordered and unordered lists of the things you love to do, to eat, etc ... Don't forget to style it a bit

Create also links to your social media profile

**Bonus:** Use image and icons to make these links more userfriendly

#### Creating Tables

As you get deeper into HTML, you will want to learn various ways to present information in a more organized way. One such way is through the use of tables.

Tables are created with the <table> tag.

* Each header in a table is specified with the <th> (“table header”) tag
* while each row is specified with the <tr> (“table row”) tag.
* The table data is then indicated with the <td> tag.

#### Example

<table>

<tr>

<th>Table Header 1</th>

<th>Table Header 2</th>

<th>Table Header 3</th>

</tr>

<tr>

<td>Input 1 under header 1</td>

<td>Input 1 under header 2</td>

<td>Input 1 under header 3</td>

</tr>

<tr>

<td>Input 2 under header 1</td>

<td>Input 2 under header 2</td>

<td>Input 2 under header 3</td>

</tr>

<tr>

<td>Input 3 under header 1</td>

<td>Input 3 under header 2</td>

<td>Input 3 under header 3</td>

</tr>

</table>

This gives us something like this:

| **Table Header 1** | **Table Header 2** | **Table Header 3** |
| --- | --- | --- |
| Input 1 under header 1 | Input 1 under header 2 | Input 1 under header 3 |
| Input 2 under header 1 | Input 2 under header 2 | Input 2 under header 3 |
| Input 3 under header 1 | Input 2 under header 2 | Input 3 under header 3 |

Here, the table is styled with borders, set horizontal or vertical alignment for the table’s content, introduce dividers, padding, etc. However, you will need to learn some CSS to do stuff like this. We’re keeping this tutorial basic and won’t get into that in this article.

#### Quotations In HTML

There are different types of quotations, and other elements represent these quotations.

For a basic quotation, here’s an example:

<p>This is a sample. And <q>Here is our quote</q></p>

This comes out as:

This is a sample. And Here is our quote

When you use the <q> tag, quotation marks are automatically added to what is enclosed within the tag.

You can also use blockquotes, which is done with the <blockquote> tag.

#### Example

<p>

<blockquote>

This is a sample.

And Here is our blockquote. In this example, we try to demonstrate how to format text to indicate a blockquote in HTML.

This is different from ordinary quotes, in that the actual "quote" symbol may or may not be added depending on

CSS styling, but the text is highlighted.

</blockquote>

</p>

It will look like this:

*This is a sample. And Here is our blockquote. In this example, we try to demonstrate how to format text to indicate a blockquote in HTML. This is different from ordinary quotes, in that the actual "quote" symbol may or may not be added depending on CSS styling, but the text is highlighted*

#### Using Comments In HTML

When doing any form of coding, it is vital to learn how to include comments in your code. Comments are used to make your code more organized. You can include a reminder to yourself or a note to others to make things easier.

Initially, it might not seem important, but when you start writing hundreds or thousands of lines of code, and when things seem to start to get complicated, comments will come in handy.

Comments won’t be shown to the viewer by the browser. Remember that they can be seen in the source code.

Comments can also be used anywhere in the code. They won’t change the function of your code in any way.

You can include a comment by opening a bracket, including a double hyphen, adding your comment, including another double hyphen, and then closing your bracket.

#### Example

<!-- This is a comment used to indicate information I want to remember or that I want others to take note of when looking at this code in the future -->

It is also worth noting that, especially when debugging, you can comment out lines of code. This way, the code remains in the document but does not function since you have commented it.

#### Example

<!-- <b>He is a boy</b> -->

While “He is a boy” is supposed to come out as bold, we have commented it, and the browser would ignore it. In this scenario, as far as the browser is concerned, we have nothing.

This could also be used this way:

<b>He <!-- is a --> boy</b>

which shows **He boy**.

This is because we commented out “is a”. We’re just trying to show you that practically anything, within or outside an element, can be commented out. You can include a note in your comment for reference – it doesn’t change the nature of what is displayed.

#### Exercise 3

Work on the HTML file you previously created

Create a table with 3 columns and 4 rows. The first row should be the headers (ie. the title of each column), The first column should display 3 projects that you would like to create here at Developers Institute, the second column should display your 3 best skills in life, and the third column should display your 3 hobbies.

**Bonus:** Try to style the table with CSS (Hint: Check out "Inline CSS Table Borders" on Google)

Display a quote that represents you the best

#### Using IFrames In HTML

In an HTML document, an iFrame (“inline frame”) can be used to embed a file in the current document. In essence, you can use it to display another website or content from another web page inside the current one.

#### Example

<iframe src="https://websitesetup.org"></iframe>

With the above code, I tell the browser to display the content of https://websitesetup.org in a frame directly inside this page.

Iframes can be used to display a website, video, images, or any other form of content.

The code for this is simple:

<iframe src="URL"></iframe>

(simply replace “URL” with the link to the website/content you want to be displayed.)

You can further customize to specify specific values, e.g., your iframe’s height and width.

#### Example

<iframe src="URL" height="350px" width="400px"></iframe>

We tell the browser to display the iframe using a height of 350px and a width of 400px.

#### HTML Forms

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

You can see HTML forms on (almost) every website, as the “sign-in” section.

Here is a simple form example:

<form>

<label>Username:</label><input type="text">

<label>Password:</label><input type="password">

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

</form>

#### Defining A Form

To define a form, use the <form></form> element.

##### The Input Element

This is the most commonly used element within HTML forms.

The <input> tag allows you to specify various user input fields, depending on the type attribute. An input element can be of type:

* text field
* password field
* checkbox
* radio button
* submit button
* reset button
* file select box

And a lot more!

To change the type of a field, pass the type="<field\_type>" as attribute in the <input> element.

You can specify the size of the field by passing it in the attributes.

##### The Label Element

This element is created to define the label of a <input> element.

### Conclusion

In conclusion, HTML is more complicated than what you have in this guide. Do not give up hope – if you can grasp the basic concepts in this guide, you are up to a good start. It’s like knowing the ABC’s of a language, and everything else becomes much easier.

If you went through this tutorial intending to create a stunning website, we’re sorry to disappoint – it just doesn’t work that way. This is a basic HTML tutorial. It’s here to help you develop a foundation you can build upon. You can be sure that it will take significantly more effort – and a lot of development time, often using a combination of languages – to create an amazing website.

## HTML5 Semantic Elements

**Done**

### Table Of Contents

* + - [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#What_you_will_learn)
    - [Useful Resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#Useful_Resources)
    - [HTML Block and Inline Elements](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#HTML_Block_and_Inline_Elements)
      * [1. Block-level elements:](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#1._Block-level_elements:)
        + [Section and Article](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#Section_and_Article)
      * [2. Inline elements](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#2._Inline_elements)
    - [3. HTML5 Semantic Elements](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#3._HTML5_Semantic_Elements)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/25#Feedback)

Last updated : March, 31st 2021

### What You Will Learn

* HTML semantic elements
* HTML block and inline elements

### Useful Resources

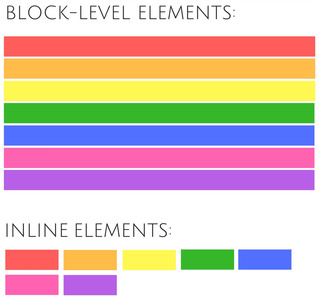
* [HTML Block and Inline Elements](https://www.w3schools.com/html/html_blocks.asp).
* [7 alternatives to the div tag](https://medium.com/@zac_heisey/7-alternatives-to-the-div-html-tag-7c888c7b5036).

HTML5 is the fifth version of the HTML scripting language. It supports a lot of new things that older versions of HTML do not.

For Example:  
In HTML5 there is something new called the Semantic Elements. Semantic elements have meaningful names which tell about the type of content: header, footer, navbar, … etc.

HTML5 introduces many semantic elements as mentioned below, making the code easier to write and understand for the developer and instructing the browser on how to treat them.

### HTML Block And Inline Elements



#### 1. Block-Level Elements:

Block-level element always starts on a new line and takes up the full width, from left to right. A block-level element can take up one line or multiple lines and has a line break before and after the element.

Example of block-level elements

* <div> : often used as a container for other HTML elements and to separate them for the rest
* <header>
* <nav>
* <article>
* <section>

I'm a div and I take the full width of a page

##### Section And Article

Both these elements are used for sectioning content. To decide which one of these you should choose, take note of the following:

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

#### 2. Inline Elements

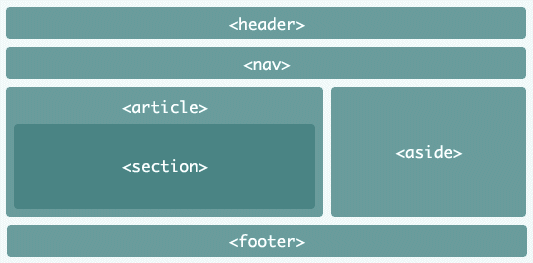
An inline element is usually used within other HTML elements. It does not cause a line break (and does not take up the full width of a page, only the space bounded by its opening and closing tag.

Example of inline elements:

* <span> : often used as a container for some text.
* <a>
* <img>

Hey, I'm a red span in a blue div

### 3. HTML5 Semantic Elements



## CSS Basics

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#What_you_will_learn)
  + [Useful Resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Useful_Resources)
    - [CSS Basics](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#CSS_Basics)
      * [1. What is CSS](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#1._What_is_CSS)
      * [2. CSS Syntax](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#2._CSS_Syntax)
      * [3. CSS Stylesheets](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#3._CSS_Stylesheets)
      * [4. CSS Simple Selectors](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#4._CSS_Simple_Selectors)
        + [Element Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Element_Selector)
        + [HTML id and class](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#HTML_id_and_class)

[Id Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Id_Selector)

[Class Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Class_Selector)

* + - * + [Universal Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Universal_Selector)
        + [Grouping Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Grouping_Selector)
      * [5. CSS Combinators Selectors](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#5._CSS_Combinators_Selectors)
        + [Descendant Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Descendant_Selector)
        + [Child Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Child_Selector)
        + [Sibling Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Sibling_Selector)
        + [Adjacent Sibling Selector](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Adjacent_Sibling_Selector)
      * [6. CSS Properties](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#6._CSS_Properties)
        + [Colors](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Colors)
        + [Backgrounds](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Backgrounds)
        + [Borders](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Borders)
        + [Margins](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Margins)
        + [Padding](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Padding)
        + [Elements dimensions](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Elements_dimensions)
        + [Text styling](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Text_styling)
        + [List styling](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#List_styling)
        + [Elements display](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/26#Elements_display)

Last updated : April, 2nd 2021

## What You Will Learn

* Starting CSS from scratch
* The concept of Colors and Backgrounds in CSS.
* Fonts and Text
* Widths and Heights in CSS
* Margin, Padding
* Floating and Display.
* Links and Lists.
* Classes IDs

## Useful Resources

* [Basic Introduction to CSS](https://medium.com/the-andela-way/basic-introduction-to-css-81f6041b92d0)
* [35+ HTML & CSS Resources for Beginners](https://medium.com/level-up-web/30-html-css-resources-for-beginners-4e4d0af4b44b)
* [Learn Layout](https://learnlayout.com/display.html)

### CSS Basics

#### 1. What Is CSS

CSS stands for **C**ascading **S**tyle **S**heets. While HTML defines the structure of a web page. CSS describes how elements are displayed on the screen. HTML represents content, and CSS represents the appearance of the content.

#### 2. CSS Syntax

A CSS rule-set consists of a selector and some declarations.

The selector points to the HTML element you want to style. It can be a html tag, class or id.

Declarations are defined in a declaration block, surrounded by curly brackets {}, each declaration includes a CSS property name and value in this format name: value, and ends with a semicolon ;.

You can also add comments, surrounded by /\* and \*/.

For example:

h1 { /\* Selector \*/

color: blue; /\* Declaration 1 \*/

font-size: 12px; /\* Declaration 2 \*/

}

#### 3. CSS Stylesheets

We can write CSS in three different places.

First, it can be written in any HTML element,

For example :

<h1 style="color:blue; font-size:12px;">Hello World</h1>

A more readable way is to write CSS in a **stylesheet**; a **stylesheet** is a text file with the .css extension. The browser will read this text file, and the webpage will apply the style.

You need to specify the path of your **stylesheet** in your HTML code if you want it to be executed:

<html>

<head>

<link rel="stylesheet" href="/path/to/stylesheet.css"></link>

</head>

<body>

<h1>Hello world</h1>

</body>

</html>

CSS can also be written directly on the HTML page, but that method is deprecated.

#### 4. CSS Simple Selectors

CSS selectors are used to “find” (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

* Simple selectors (select elements based on name, id, class)
* [Combinator selectors](https://www.w3schools.com/css/css_combinators.asp) : Select elements based on a specific relationship between them
* [Pseudo-class selectors](https://www.w3schools.com/css/css_pseudo_classes.asp) : Select elements based on a certain state
* [Pseudo-elements selectors](https://www.w3schools.com/css/css_pseudo_elements.asp) : Select and style a part of an element
* [Attribute selectors](https://www.w3schools.com/css/css_attribute_selectors.asp): Select elements based on an attribute or attribute value)

##### Element Selector

Element selector selects HTML elements based on the element name (tag).

For example, this style will be applied on every <p> tag:

p {

color: red;

}

##### HTML Id And Class

HTML elements can have a class attribute, used to define similar styles for elements with the same class name. Elements can also have a unique id attribute.

###### Id Selector

The **id selector** uses the id attribute of an HTML element. To use it, write a hash **#** character, followed by the element’s id.

For example, to style this paragraph <p id="welcome">Hello World</p>, use:

#welcome{

color: red;

}

###### Class Selector

While the **class selector** uses the class attribute of the elements, to use it, write a period **.** followed by the class of the element.

For example, to style this image <img class="netfliximg" src="netflix-pic"/>, use:

.netfliximg{

width: 200px;

height: 200px;

}

##### Universal Selector

**Universal selector** is also available in CSS (and in a lot of computer languages), **\*** means “Everything.”

This style will be applied on every HTML elements of the page:

\* {

color: blue;

}

##### Grouping Selector

You can style more than one element at the time; just separate each selector with a comma **,**:

h1, h2, p {

color: red;

}

#### 5. CSS Combinators Selectors

##### Descendant Selector

The descendant selector matches **all descendants** of a specified element (meaning they are nested in it).

The following example selects all <p> elements inside <div> elements:

div p {

background-color: yellow;

}

##### Child Selector

The child selector selects all elements that are the **direct descendant** of a specified element.

The following example selects all <p> elements that are children of a <div> element:

div > p {

background-color: yellow;

}

##### Sibling Selector

The general **sibling selector** selects all elements that are siblings of a specified element.

Sibling means “that have the same parent”.

The following example selects all <p> elements that are siblings of <div> elements:

div ~ p {

background-color: yellow;

}

##### Adjacent Sibling Selector

The **adjacent sibling selector** selects all elements that are the adjacent siblings of a specified element.

“Adjacent” means “immediately following”.

The following example selects all <p> elements that are placed immediately after <div> elements:

div + p {

background-color: yellow;

}

##### Exercise 1

Create a stuctured HTML file

Copy this paragraph and style it with Inline CSS first and then with a CSS stylesheet.

*Hello Everyone, Congratulation on your first day of coding at DevelopersInstitute*

What happens if you style this paragraph both with Inline CSS and with a CSS stylesheet ?

#### 6. CSS Properties

##### Colors

In CSS, colors can be Hexadecimal Color Codes (#ffffff), rgb (rgb(255,255,255)), or color names (black).

##### Backgrounds

You can use CSS to add a background to your sections; it can be a background-color or a background-image.

background-image needs a URL to work.

body{

background-image: url("my\_img.png")

}

##### Borders

The CSS border properties allow you to specify the border-style, border-width, and border-color of an element’s border.

p {

border-style: solid;

border-width: 5px;

border-color: green;

}

##### Margins

The CSS margin properties are used to create space **around elements**, **outside** of any defined borders.

You can specify the width of margin for every side, but you can also set the margin for each element’s side.

p {

margin-top: 100px;

margin-bottom: 100px;

margin-right: 150px;

margin-left: 80px;

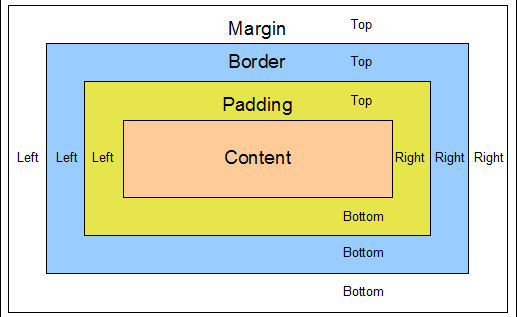
}

##### Padding

The CSS padding properties are used to generate space **around an element’s content**, **inside** of any defined borders.

It works the same way as margin.

See the [box model](https://www.w3schools.com/css/css_boxmodel.asp) to understand padding and margin



##### Elements Dimensions

The height and width properties are used to set an element’s height and width.

##### Text Styling

You can specify the color of the text, but also the alignment (with text-align), the text-decoration, the text-transformation, the text-indent and [much more](https://www.w3schools.com/css/css_text.asp)

p {

color: blue;

text-align: justify;

text-indent: 50px;

text-decoration: underline;

text-transform: uppercase;

}

##### List Styling

You can modify the icon in front of each list elements, with list-style-type to use a built in icon, or with list-style-image to add an image of your own.

##### Exercise 2

On the same HTML file

Recreate the green square of the Exercise 1, using margin, padding and borders

Try different types of borders, and shadows (Check it on Google)

##### Elements Display

You can modify the way elements are displayed.

The display property is controlling the layout of an element.

* Block-level: display: block; will display the element as a block, meaning stretch it out to the left and right as far as possible, and the next element will be placed below it.
* Inline: display: inline; elements don’t start on a new line and only take up as much width as necessary.
* Inline block: display: inline-block allows to set a width and height on the element and respect the padding, but doesn’t add a line break after the element.
* None: display: none won’t display the element. **Careful** : It will not hide it (meaning it won’t keep a blank place for the element). To hide an element, use visibility: hidden

See  
[This Course on CSS Layout](https://www.w3schools.com/css/css_display_visibility.asp)  
[Example with squares](https://www.w3schools.com/css/tryit.asp?filename=trycss_inline-block_span1)  
[Example within texts](https://www.w3schools.com/cssref/tryit.asp?filename=trycss_display)

##### Exercise 3

On the same HTML file

Recreate this flag with the **display property**



You can look at these links to help you :

* [css-display-property](https://www.geeksforgeeks.org/css-display-property/)
* [Understanding CSS Display: None, Block, Inline and Inline-Block](https://medium.com/better-programming/understanding-css-display-none-block-inline-and-inline-block-63f6510df93)

##### Exercise 5

On the same HTML file

Copy this code and do the following steps :

<div>

<ul>

<li>What is Lorem Ipsum?</li>

<li>Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, </li>

<li>when an unknown printer took a galley of type and scrambled it to make a type specimen book. </li>

</ul>

</div>

<div>

<p>It has survived not only five centuries, but also the leap into electronic typesetting,</p>

<ol>

<li> remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets </li>

</ol>

<p> containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.</p>

</div>

1.Change the second li of the 1st div to blue

2.Make all the li of the first div in uppercase and in bold

3.Change the 1st paragraph of the second div so its background color will be pink and its color white

4.Make the 2nd paragraph of the second div with a background color light blue, with padding and margin

**Bonus:** Try to find a few ways to achieve each of the steps

## CSS Advanced

**Done**

### Table Of Contents

* + [What you will learn](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#What_you_will_learn)
  + [Useful Resources](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#Useful_Resources)
    - [Elements position](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#Elements_position)
    - [Dealing with overflow](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#Dealing_with_overflow)
    - [CSS Pseudo Class Selectors](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#CSS_Pseudo_Class_Selectors)
      * [hover](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#hover)
      * [children](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#children)
      * [not](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#not)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1140#Feedback)

Last updated : April, 2nd 2021

## What You Will Learn

* Positions
* Overflow
* Pseudo Class Selector

## Useful Resources

* [Learn Layout](https://learnlayout.com/display.html)

### Elements Position

The position property specifies the type of positioning method used for an element (static on the page, relative to his parent, etc..)

It can be:

* static: Default value. Elements render in order, as they appear in the document flow
* relative: The element is positioned relative to its normal position; you can add left, right, top or bottom arguments to add a gap by the element.
* fixed: Always stays in the same place even if the page is scrolled, left, right, top and bottom are used to position the element
* absolute: The element is positioned relative to its first positioned (not static) ancestor element
* sticky: The element is positioned based on the user’s scroll position

[More on CSS Position](https://www.w3schools.com/css/css_positioning.asp)  
[CSS Position Tricks](https://css-tricks.com/almanac/properties/p/position/)

#### Exercise 1

On the same HTML file (as the one in the CSS lesson)

Recreate this image with the **position** property

The elements are : a parent box (grey), a blue box and an orange box



### Dealing With Overflow

The overflow property specifies whether to clip the content or add scrollbars when an element’s content is too big to fit in the specified area.

It can be:

* visible - Default. The overflow is not clipped. The content renders outside the element’s box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

[Look at the demonstration Here](https://www.w3schools.com/cssref/playit.asp?filename=playcss_overflow)

### CSS Pseudo Class Selectors

A pseudo-class is used to style an element when he is in a special state (for example, when the user hovers over it)

The syntax is : element:pseudo-class {...}

#### Hover

hover means “mouse is over it”, for example, to change the color of a <div> when the mouse goes over it:

div:hover {

background-color: blue;

}

#### Children

You can target elements only if they are the nth child of their parent. For example, to style <p> tags that are the first child of another element, use:

p:first-child{

color: blue;

}

The same thing works with last-child or nth-child(n)

#### Not

The not pseudo-class allows you to exclude elements; for example, select all the divs that are not of the ignore class.

div:not(.ignore){

background-color: yellow;

}

For more, see [this](https://www.w3schools.com/css/css_pseudo_classes.asp).

## Basic HTML5 Template For Any Project

**Done**

### Table Of Contents

* + - [DOCTYPE](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#DOCTYPE)
    - [HTML element](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#HTML_element)
    - [HEAD element](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#HEAD_element)
    - [Fair Competition](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#Fair_Competition)
      * [Next is history](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#Next_is_history)
      * [Next step](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#Next_step)
* [Feedback](http://learn.di-learning.com/courses/collection/6/course/9/section/14/chapter/1126#Feedback)

Last updated : April, 2nd 2021

As you learn HTML5 and add new technologies to your toolbox, you may want to build a boilerplate for yourself to start all HTML5-based projects. We encourage this, and you can also consider using one of the online resources, which gives you an essential starting point for HTML5.

In this article, we will see how to get started. Let’s start with a simple, basic HTML5 page:

<!doctype html>

<html>

<head>

<meta charset = "utf-8">

<title> The HTML5 Herald </ title>

<meta name = "description" content = "The HTML5 Herald">

<meta name = "author" content = "SitePoint">

<link rel = "stylesheet" href = "css/styles.css?v = 1.0">

</head>

<body>

<script src = "js/scripts.js"></script>

</body>

</html>

With the basic template in place, let’s examine some of the markup’s important parts and how they differ from the way HTML was written before HTML5.

### DOCTYPE

First, we have a “document type declaration” or doctype. This is just one way to tell the browser (or any other parser) what type of document is being viewed. For HTML files, it represents a specific version and style of HTML.

The doctype should always be the first item at the top of any HTML file. Many years ago, the doctype declaration was ugly and hard to remember mess. For XHTML 1.0 Strict:

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

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

HTML4 conversion:

<! DOCTYPE HTML PUBLIC "-// W3C // DTD HTML 4.01 Transitional // EN"

"http://www.w3.org/TR/html4/loose.dtd">

Although the long string of text at the top of the document didn’t hurt us (apart from forcing visitors to our website to download some extra bytes), HTML5 has eliminated the obscure things. What you need now is:

<! doctype html>

Simple and fair. Doctype can be written in uppercase, lowercase, or mixed case. You will notice that the “5” is missing from the statement. Although the current iteration of web markup is called “HTML5”, it is really just an evolution of the previous HTML standard-future specifications will only be an evolution of what we have today.

Since browsers are often required to support all existing content on the web, there is no need to rely on document types to tell them which features a given document should support. In other words, doctype alone will not make your page compatible with HTML5. It all depends on the browser.

You can use one of these two old document types with new HTML5 elements on the page, and the page will look the same as when using the new doctype.

### HTML Element

The following HTML document is an HTML element, which has not changed significantly in HTML5. In our example, we set lang the attribute’s value to en, which specifies that the document is in English. In the based xhtmlsyntax, xmlns attributes need to be included. In HTML5, this is no longer necessary, and even lang attributes are not necessary for the validation or correct operation of the document.

This is what we have so far, including closed </html> tags:

<! doctype html>

<html lang = "en">

</ html>

### HEAD Element

The next part of the page is the <head> section. head The first line defines the document’s character encoding. This is another element that has been simplified since XHTML and HTML4 and is an optional feature, but recommended. In the past, you could write:

<meta http-equiv = "Content-Type" content = "text / html; charset = utf-8">

HTML5 improves this by minimizing character encoding tags:

<meta charset = "utf-8">

In almost all cases, UTF-8 is the value you will use in the document. A complete explanation of character encoding is beyond this article’s scope, and you may not be interested in it. Nonetheless, if you want to dig deeper, you can read topics about [W3C](https://html.spec.whatwg.org/multipage/infrastructure.html#encoding-terminology) or  
[WHATWG](https://html.spec.whatwg.org/multipage/infrastructure.html#encoding-terminology) .

**Note**: To ensure that all browsers can read character encoding correctly, the encoding declaration must be included within the first 512 characters of the document. It should also appear before any content-based elements (such as the <title> element that follows it in our sample site).

We still have a lot to write about this topic, but we want to keep you awake-so we won’t tell you those details! Now, we are content to accept this simplified statement and then move on to the next part of the document:

<title> The HTML5 Herald </ title>

<meta name = "description" content = "The HTML5 Herald">

<meta name = "author" content = "SitePoint">

<link rel = "stylesheet" href = "css/styles.css? v = 1.0">

In these lines, HTML5 is almost indistinguishable from the previous syntax. The page title’s declaration (the only required element in the header) is the same as before. The meta tags we include are just some optional examples to indicate these tags’ position; you can put as many valid meta elements here.

This markup block’s key part is the style sheet, which is contained using the idiomatic link element. Except href and rel, link does not require other attributes. The type attribute (common in older versions of HTML) is not required and is not required to indicate the style sheet’s content type.

### Fair Competition

When HTML5 was introduced, it included many new elements, such as articles and sections. You may think that this is the main problem with the support of unrecognized elements in older browsers, but you’re mistaken. This is because most browsers don’t actually care what tags you use.

If you have an HTML document with a recipe tag (or even a ziggy tag), and your CSS attaches some styles to the element, almost every browser will treat it as a completely regular operation; nothing Apply your style without complaints.

Of course, such hypothetical documents will not be verifiable, and there may be accessibility issues. Still, it will render correctly in almost all browsers-except for older versions of Internet Explorer (IE). Before version 9, IE prevented unrecognized elements from receiving styles.

These mysterious elements are considered “unknown elements” by the rendering engine, so you cannot change their appearance or behavior. This includes the elements we imagine and any elements that were not defined when the browser version was developed. This means new HTML5 elements.

The good news is that the usage of IE has dropped. The global usage of IE11 has dropped to about 2.7% (as of 2018), and the previous version has almost disappeared from the map.

However, if you need to support ancient browsers, you can still use the trusted HTML5 Shiv, a straightforward JavaScript initially developed by John Resig. Inspired by Sjoerd Visscher’s idea, it enables new HTML5 elements to be styled in older versions of IE.

However, in reality, it is not needed now. All modern browsers and even the latest older versions support HTML5 elements. The one exception is that some browsers do not recognize the newer significant elements. However, for these browsers, you can still use this element as long as you add the appropriate style (such as setting it as a block element).

#### Next Is History

Looking at the rest of the start template, we have common body elements with their end tags and closing html tags. We also reference a JavaScript file in the script element.

Much like the link tag discussed earlier, the <script> tag does not need to declare a type attribute. If you have ever written XHTML, you may remember that your script tags look like this:

<script src = "js/scripts.js" type = "text/javascript"></script>

Since JavaScript is the only true scripting language used on the Web, and all browsers assume that you are using JavaScript, even if you do not explicitly state the fact, the type attribute is unnecessary in HTML5 documents:

<script src = "js/scripts.js"></script>

We place the script element at the bottom of the page to comply with best practices for embedding JavaScript. This has to do with page load speed; when the browser encounters a script, it will pause the download and render the rest of the page while parsing it.

This will cause the page to load much slower when including large scripts at the top of the page before loading anything. That’s why most scripts should be placed at the very bottom of the page so that they will only be parsed after the rest of the page is loaded.

However, in some cases (such as using HTML5 shiv), the script may need to be placed at the head of the document because you want to take effect before the browser starts rendering the page.

#### Next Step

One way to take HTML5 to the next level is to try the HTML5 Boilerplate. This regularly updated resource provides a convenient starting point for your project, with all the latest best practices established by hundreds of the best programmers in the world.

Even if you want to go through the code and see how certain elements are used these days, such as the various meta-elements in the document header, it’s worth downloading and checking out.

Another way to take your website or web application development to the next level is to try a modern framework that is widely used today.

The above is the detailed content of the basic HTML5 template suitable for any project.