What is HTML

HTML is an acronym which stands for **Hyper Text Markup Language** which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. **With the help of HTML only, we can create static web pages**.

Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

Let's see a simple example of HTML.

```
1. <!DOCTYPE>
2. <html>
3. <head>
4. <title>Web page title</title>
5. </head>
6. <body>
7. <h1>Write Your First Heading</h1>
8. <p>Write Your First Paragraph.</p>
9. </body>
10. </html>
```

Description of HTML Example

<!DOCTYPE>: It defines the document type or it instruct the browser about the version of HTML.

<html > :This tag informs the browser that it is an HTML document. Text between html tag describes the web document. It is a container for all other elements of HTML except <!DOCTYPE>

<head>: It should be the first element inside the <html> element, which contains the metadata(information about the document). It must be closed before the body tag opens.

<title>: As its name suggested, it is used to add title of that HTML page which appears at the top of the browser window. It must be placed inside the head tag and should close immediately. (Optional)

<body>: Text between body tag describes the body content of the page that is visible to the end user. This tag contains the main content of the HTML document.

<h1> : Text between <h1> tag describes the first level heading of the webpage.

<p>: Text between <p> tag describes the paragraph of the webpage.

Brief History of HTML

In the late 1980's , a physicist, Tim Berners-Lee who was a contractor at CERN, proposed a system for CERN researchers. In 1989, he wrote a memo proposing an internet based hypertext system.

Tim Berners-Lee is known as the father of HTML. The first available description of HTML was a document called "HTML Tags" proposed by Tim in late 1991. The latest version of HTML is HTML5, which we will learn later in this tutorial.

HTML Versions

Since the time HTML was invented there are lots of HTML versions in market, the brief introduction about the HTML version is given below:

HTML 1.0: The first version of HTML was 1.0, which was the barebones version of HTML language, and it was released in1991.

HTML 2.0: This was the next version which was released in 1995, and it was standard language version for website design. HTML 2.0 was able to support extra features such as form-based file upload, form elements such as text box, option button, etc.

HTML 3.2: HTML 3.2 version was published by W3C in early 1997. This version was capable of creating tables and providing support for extra options for form elements. It can also support a web page with complex mathematical equations. It became an official standard for any browser till January 1997. Today it is practically supported by most of the browsers.

HTML 4.01: HTML 4.01 version was released on December 1999, and it is a very stable version of HTML language. This version is the current official standard, and it provides added support for stylesheets (CSS) and scripting ability for various multimedia elements.

HTML5 : HTML5 is the newest version of HyperText Markup language. The first draft of this version was announced in January 2008. There are two major organizations one is W3C (World Wide Web Consortium), and another one is WHATWG(Web Hypertext Application Technology Working Group) which are involved in the development of HTML 5 version, and still, it is under development.

Features of HTML

1) It is a very **easy and simple language**. It can be easily understood and modified.

2) It is very easy to make an **effective presentation** with HTML because it has a lot of formatting tags.

3) It is a **markup language**, so it provides a flexible way to design web pages along with the text.

4) It facilitates programmers to add a **link** on the web pages (by html anchor tag), so it enhances the interest of browsing of the user.

5) It is **platform-independent** because it can be displayed on any platform like Windows, Linux, and Macintosh, etc.

6) It facilitates the programmer to add **Graphics, Videos, and Sound** to the web pages which makes it more attractive and interactive.

7) HTML is a case-insensitive language, which means we can use tags either in lower-case or upper-case.

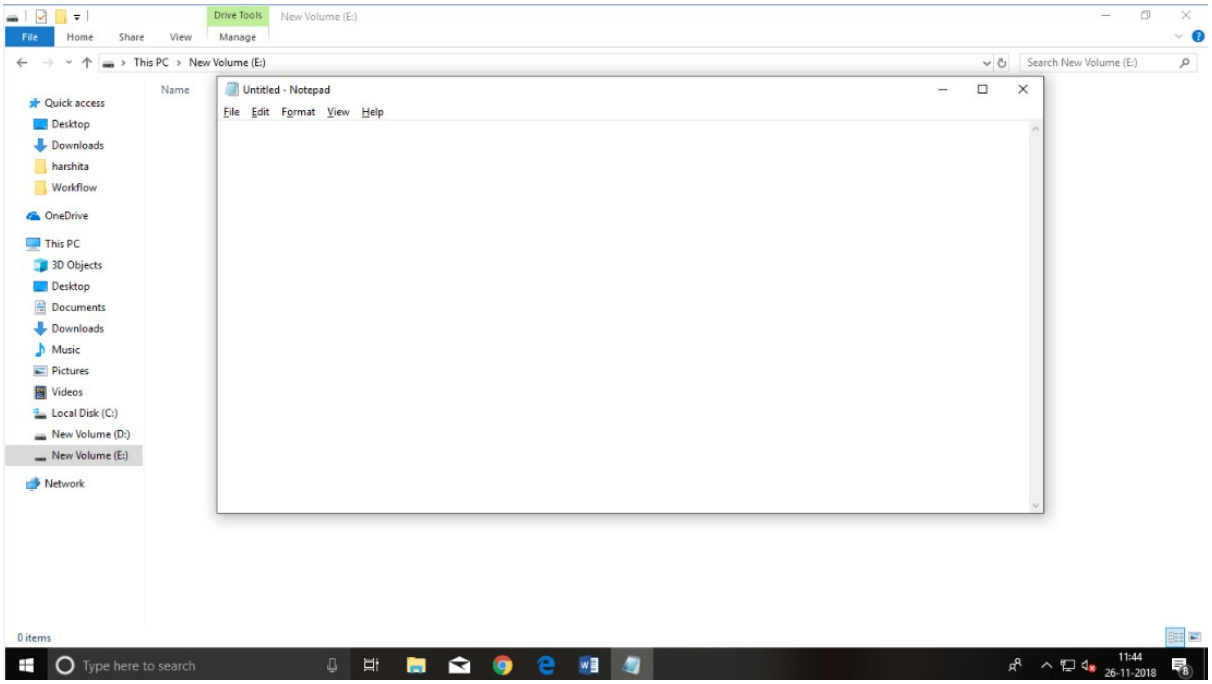
HTML text Editors

- An HTML file is a text file, so to create an HTML file we can use any text editors.
- Text editors are the programs which allow editing in a written text, hence to create a web page we need to write our code in some text editor.
- There are various types of text editors available which you can directly download, but for a beginner, the best text editor is Notepad (Windows) or TextEdit (Mac).
- After learning the basics, you can easily use other professional text editors which are, **Notepad++, Sublime Text, Vim, etc**.
- In our tutorial, we will use Notepad and sublime text editor. Following are some easy ways to create your first web page with Notepad, and sublime text.

A. HTML code with Notepad. (Recommended for Beginners)

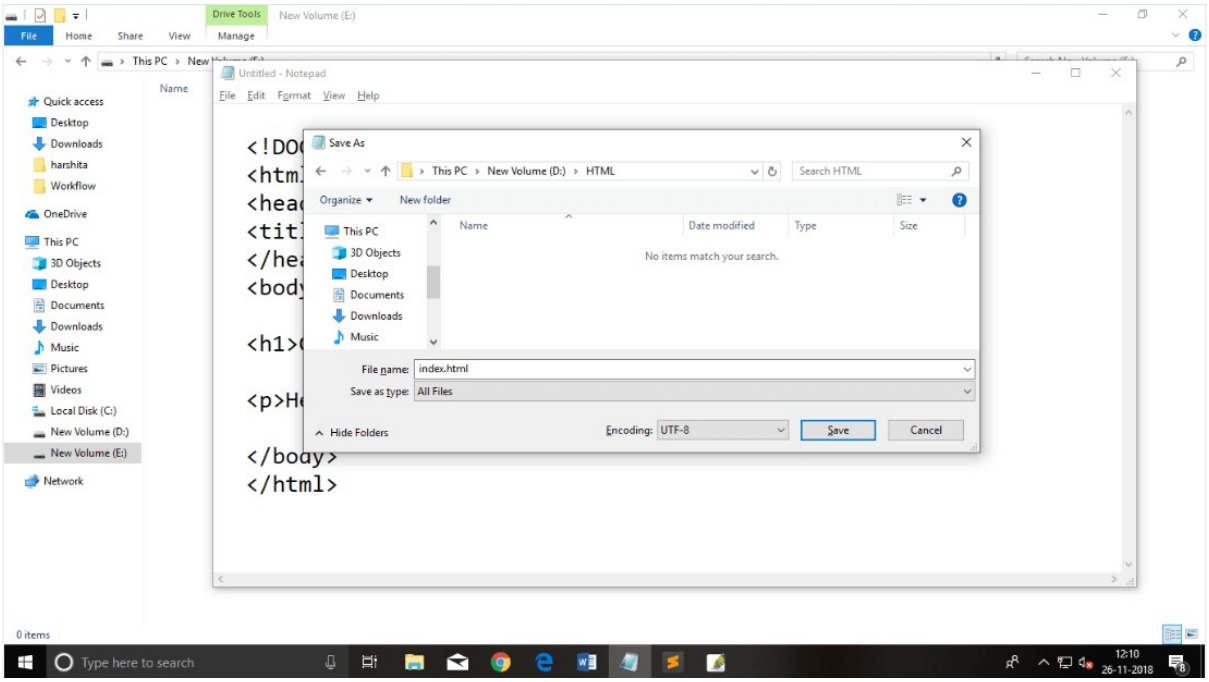
Notepad is a simple text editor and suitable for beginners to learn HTML. It is available in all versions of Windows, from where you easily access it.

Step 1: Open Notepad (Windows)



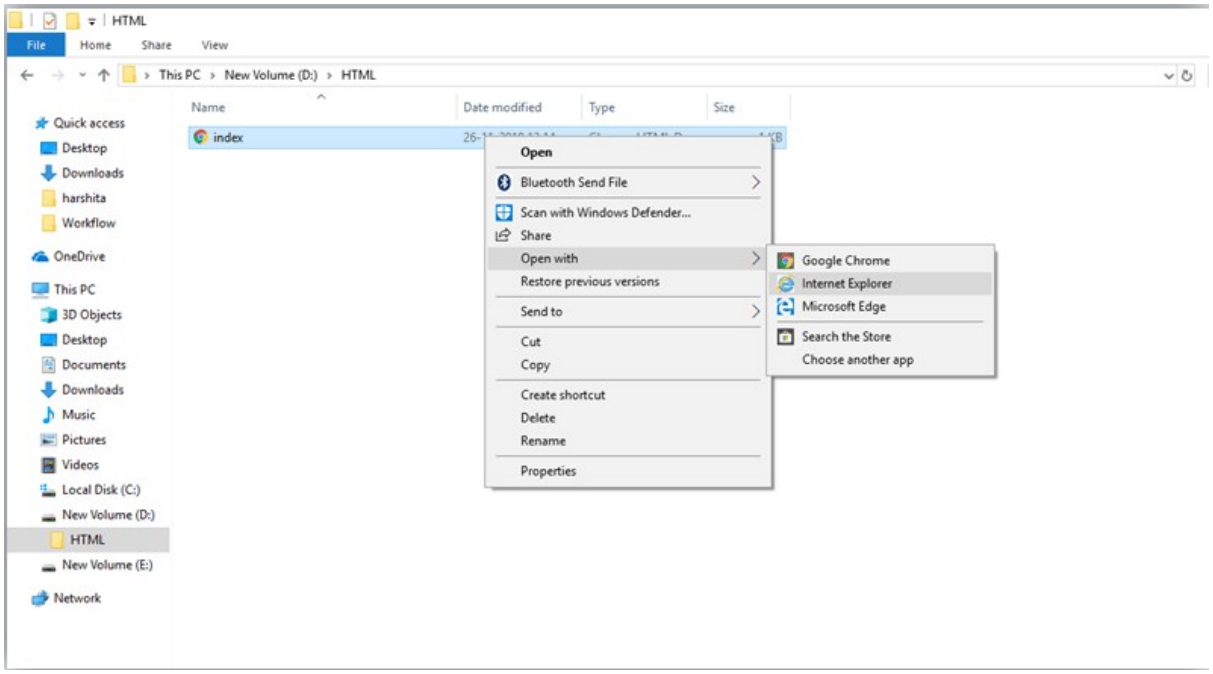
Step 2: Write code in HTML

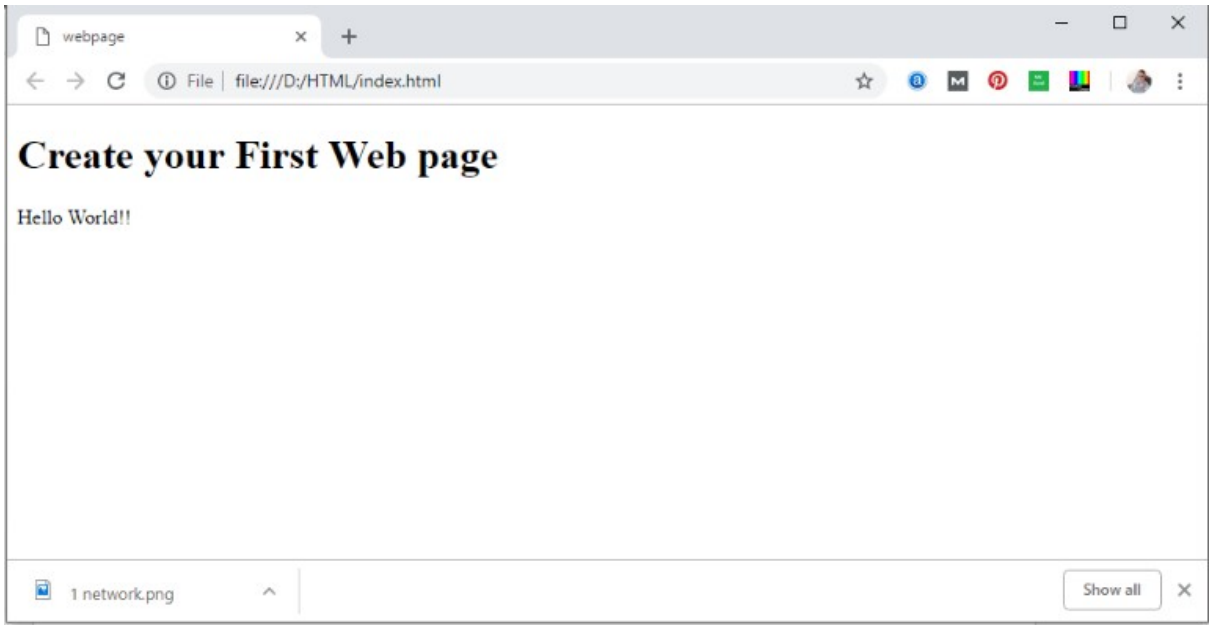
Step 3: Save the HTML file with .htm or .html extension.



Step 4: Open the HTML page in your web browser.

To run the HTML page, you need to open the file location, where you have saved the file and then either double-click on file or click on open with option



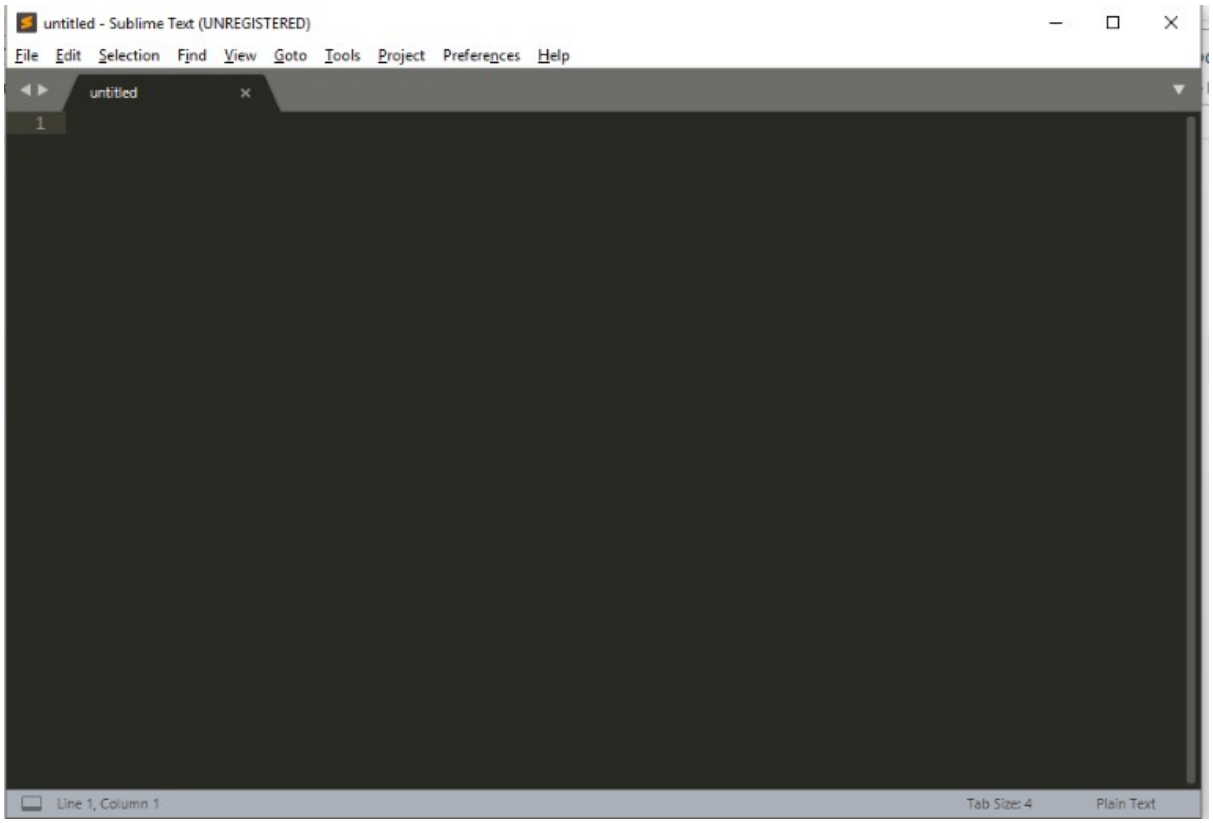


B. HTML code with Sublime Text-editor.(Recommended after learning basics of HTML)

When you will learn the basics of HTML, then you can use some professional text editors, which will help you to write an efficient and fast code. So to use Sublime Text editors, first it needs to download and install from internet. You can easily download it from this https://www.sublimetext.com/download link and can install in your PC. When installation of Sublime text editor done then you can follow the simple steps to use it:

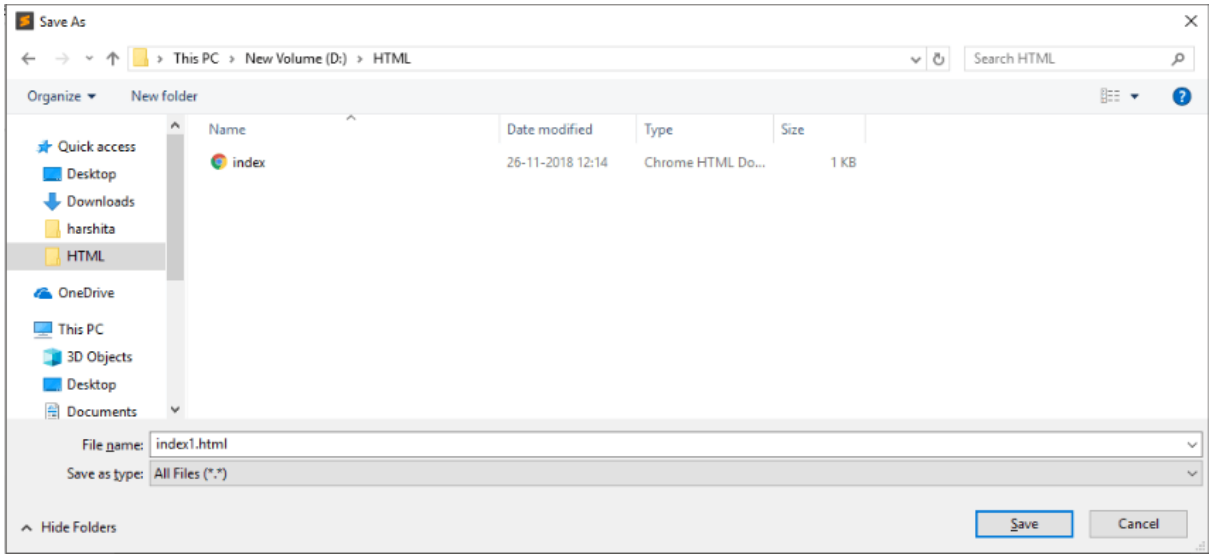
Step 1: Open Sublime Text editor(Windows 8):

To open Sublime Text editor go to **Start screen --› type Sublime Text--› Open** it. To open a new page press **CTRL+N.**

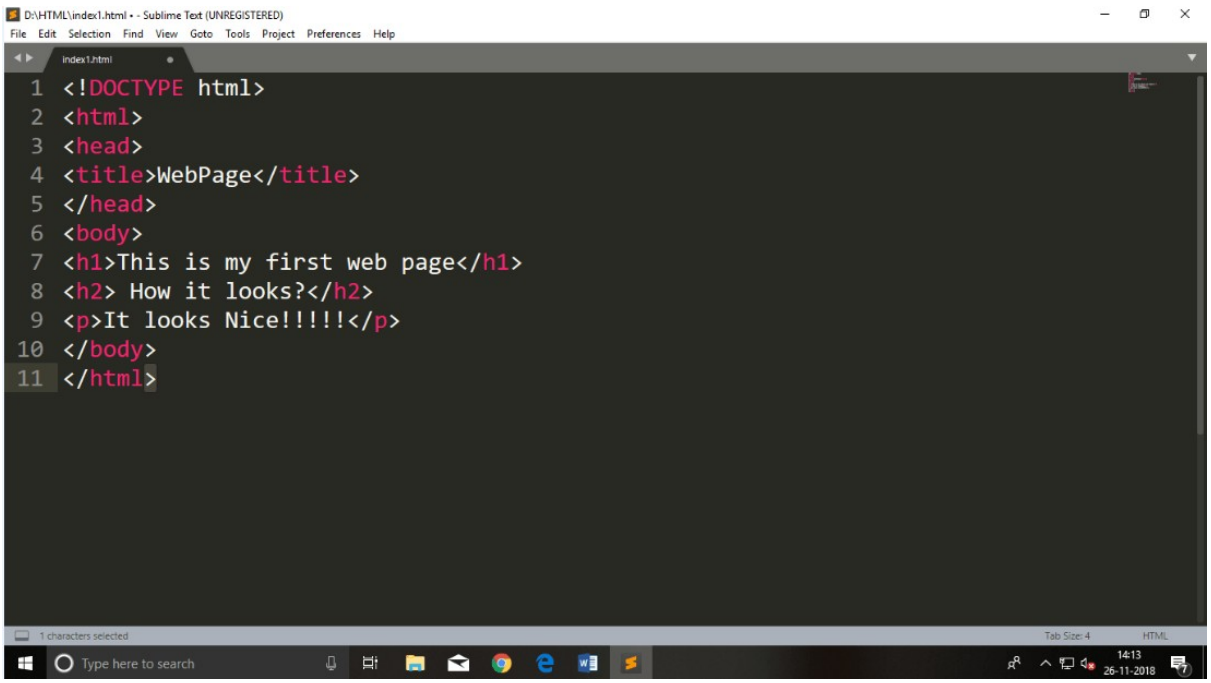


Step 2: Save the page before writing any code.

To save your page in Sublime Text press Ctrl+S or go to File option --› save, to save a file use extension .htm or .html. We recommend to save the file first then write the code because after saving the page sublime text editor will give you suggestions to write code.

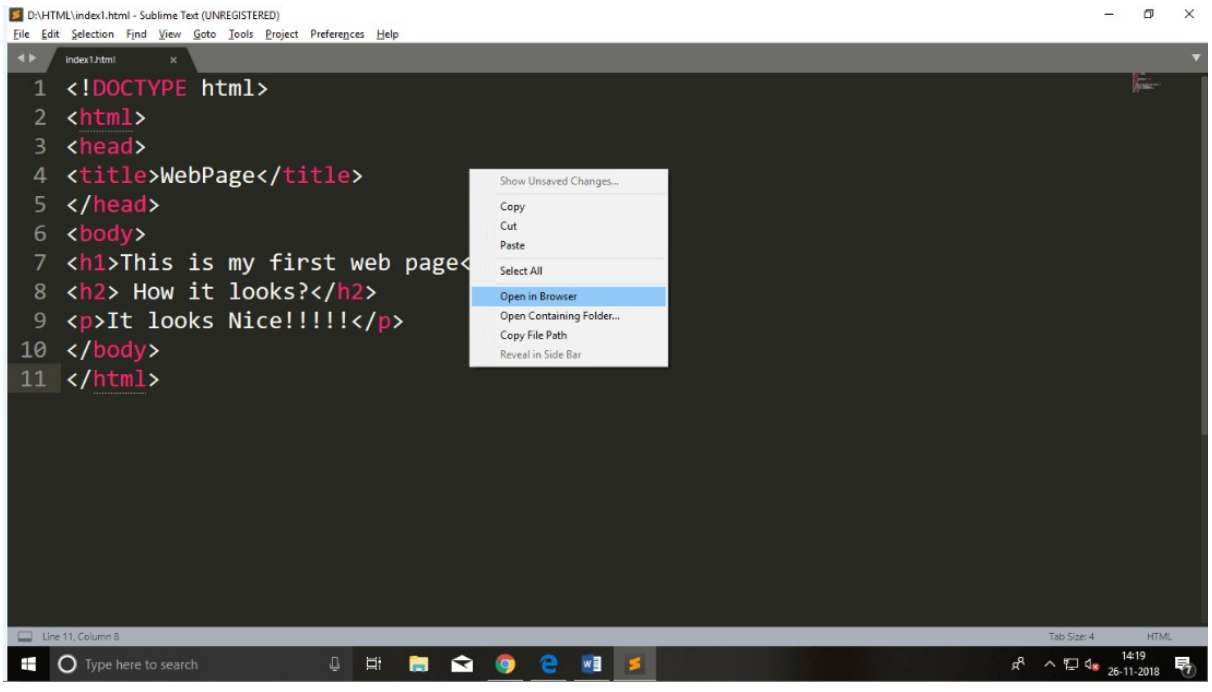


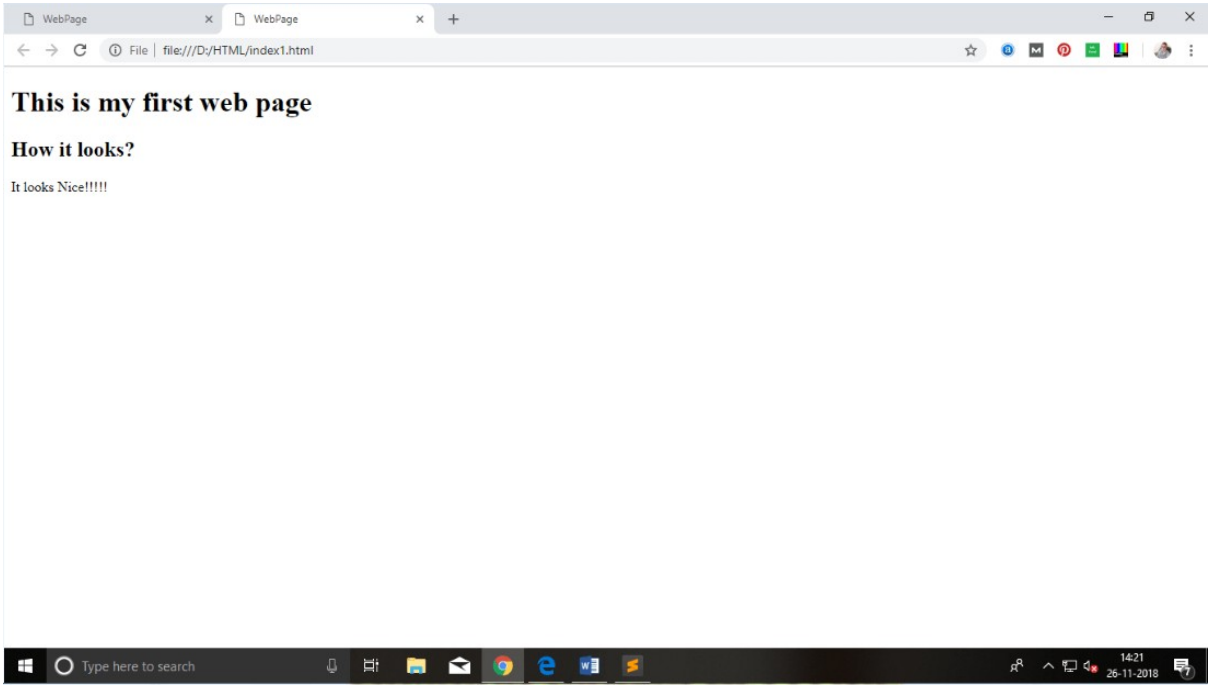
Step 3: Write the code in Sublime Text editor



Step 4: Open the HTML page in your Browser

To execute or open this page in Web browser just **right click** by mouse on sublime text page and click on **Open in Browser**.





Building blocks of HTML

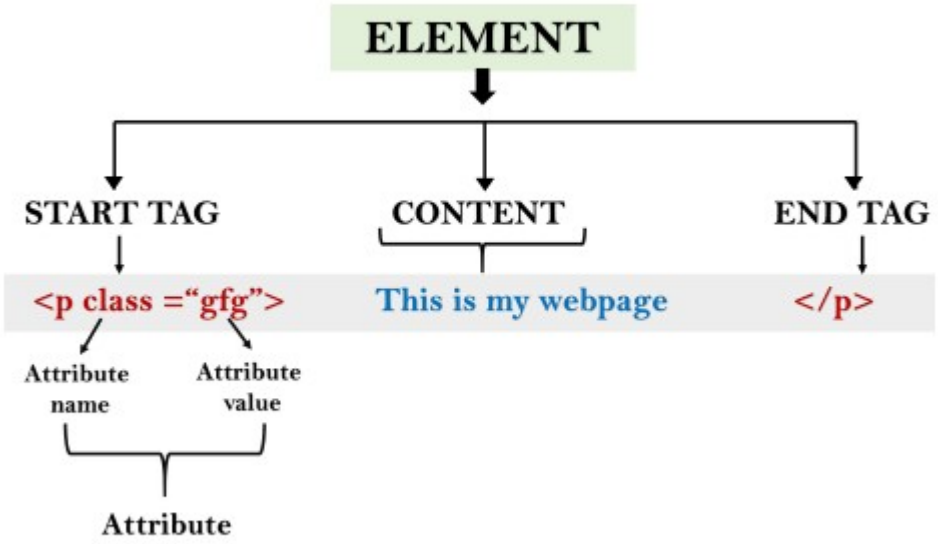
An HTML document consist of its basic building blocks which are:

- **Tags:** An HTML tag surrounds the content and apply meaning to it. It is written between < and > brackets.
- **Attribute:** An attribute in HTML provides extra information about the element, and it is applied within the start tag. An HTML attribute contains two fields: name & value.

Syntax

1. <tag name attribute_name= " attr_value"> content </ tag name>

- **Elements:** An HTML element is an individual component of an HTML file. In an HTML file, everything written within tags are termed as HTML elements.



Example:

```
1.  <!DOCTYPE html>
2.  <html>
3.    <head>
4.      <title>The basic building blocks of HTML</title>
5.   </head>
6.    <body>
7.        <h2>The building blocks</h2>
8.        <p>This is a paragraph tag</p>
9.        <p style="color: red">The style is attribute of paragraph tag</p>
10.       <span>The element contains tag, attribute and content</span>
11.  </body>
12. </html>
```

Output:

The building blocks

This is a paragraph tag

HTML Tags

HTML tags are like keywords which defines that how web browser will format and display the content. With the help of tags, a web browser can distinguish between an HTML content and a simple content. HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags.

When a web browser reads an HTML document, browser reads it from top to bottom and left to right. HTML tags are used to create HTML documents and render their properties. Each HTML tags have different properties.

An HTML file must have some essential tags so that web browser can differentiate between a simple text and HTML text. You can use as many tags you want as per your code requirement.

- All HTML tags must enclosed within <> these brackets.
- Every tag in HTML perform different tasks.
- If you have used an open tag <tag>, then you must use a close tag </tag> (except some tags)

Syntax

<tag> content </tag>

HTML Tag Examples

Note: HTML Tags are always written in lowercase letters. The basic HTML tags are given below:

 Paragraph Tag

<h2> Heading Tag </h2>

Bold Tag

Italic Tag

<u> Underline Tag</u>

Unclosed HTML Tags

Some HTML tags are not closed, for example br and hr.

**
 Tag**: br stands for break line, it breaks the line of the code.

<hr> Tag: hr stands for Horizontal Rule. This tag is used to put a line across the webpage.

HTML Meta Tags

DOCTYPE, title, link, meta and style

HTML Text Tags

<p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, , , <abbr>, <acronym>, <address>, <bdo>, <blockquote>, <cite>, <q>, <code>, <ins>, <del>, <dfn>, <kbd>, <pre>, <samp>, <var> and

HTML Link Tags

<a> and <base>

HTML Image and Object Tags

<img>, <area>, <map>, <param> and <object>

HTML List Tags

, <ol>, <li>, <dl>, <dt> and <dd>

HTML Table Tags

table, tr, td, th, tbody, thead, tfoot, col, colgroup and caption

HTML Form Tags

form, input, textarea, select, option, optgroup, button, label, fieldset and legend

HTML Scripting Tags

script and noscript

Note: We will see examples using these tags in later charters.

HTML Tags List

Following is the complete list of HTML tags with the description which are arranged alphabetically.

HTML Attribute

- HTML attributes are special words which provide additional information about the elements or attributes are the modifier of the HTML element.
- Each element or tag can have attributes, which defines the behaviour of that element.
- Attributes should always be applied with start tag.
- The Attribute should always be applied with its name and value pair.
- The Attributes name and values are case sensitive, and it is recommended by W3C that it should be written in Lowercase only.
- You can add multiple attributes in one HTML element, but need to give space between two attributes.

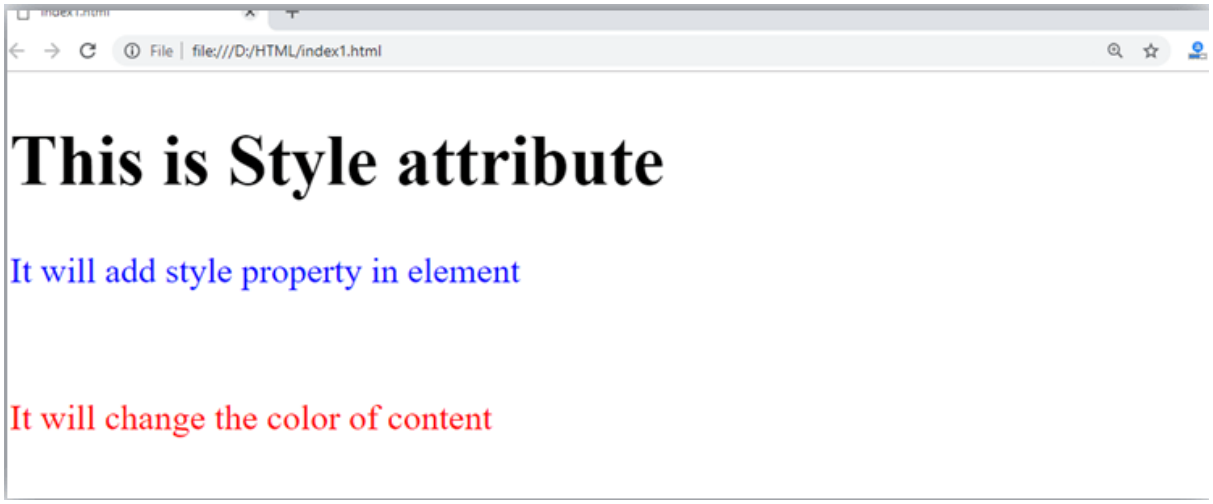
Syntax

1. <element attribute_name="value">content</element>

Example

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  </head>
5.  <body>
6.      <h1> This is Style attribute</h1>
7.     <p style="height: 50px; color: blue">It will add style property in element</p>
8.      <p style="color: red">It will change the color of content</p>
9.  </body>
10. </html>
```

Output:



Explanation of above example:

1. It will add style property in element

In the above statement, we have used paragraph tags in which we have applied style attribute. This attribute is used for applying CSS property on any HTML element. It provides height to paragraph element of 50px and turns it colour to blue.

1. It will change the color of content

In the above statement we have again used style attribute in paragraph tag, which turns its colour red.

Note: There are some commonly used attributes are given below, and the complete list and explanation of all attributes are given in HTML attributes List.

The title attribute in HTML

Description: The title attribute is used as text tooltip in most of the browsers. It display its text when user move the cursor over a link or any text. You can use it with any text or link to show the description about that link or text. In our example, we are taking this with paragraph tag and heading tag.

Example

With <h1> tag:

1. <h1 title="This is heading tag">Example of title attribute</h1>

With tag:

1. Move the cursor over the heading and paragraph, and you will see a description as a tooltip

Code:

```
<!DOCTYPE html>
<html>
  <head>
 </head>
<body>

  <h1 title="This is heading tag">Example of title attribute</h1>
  <p title="This is paragraph tag">Move the cursor over the heading and paragraph, and you will see a description as a tooltip</p>

</body>
</html>
```

Output:

The href attribute in HTML

Description: The href attribute is the main attribute of <a> anchor tag. This attribute gives the link address which is specified in that link. **The href attribute provides the hyperlink, and if it is blank, then it will remain in same page**.

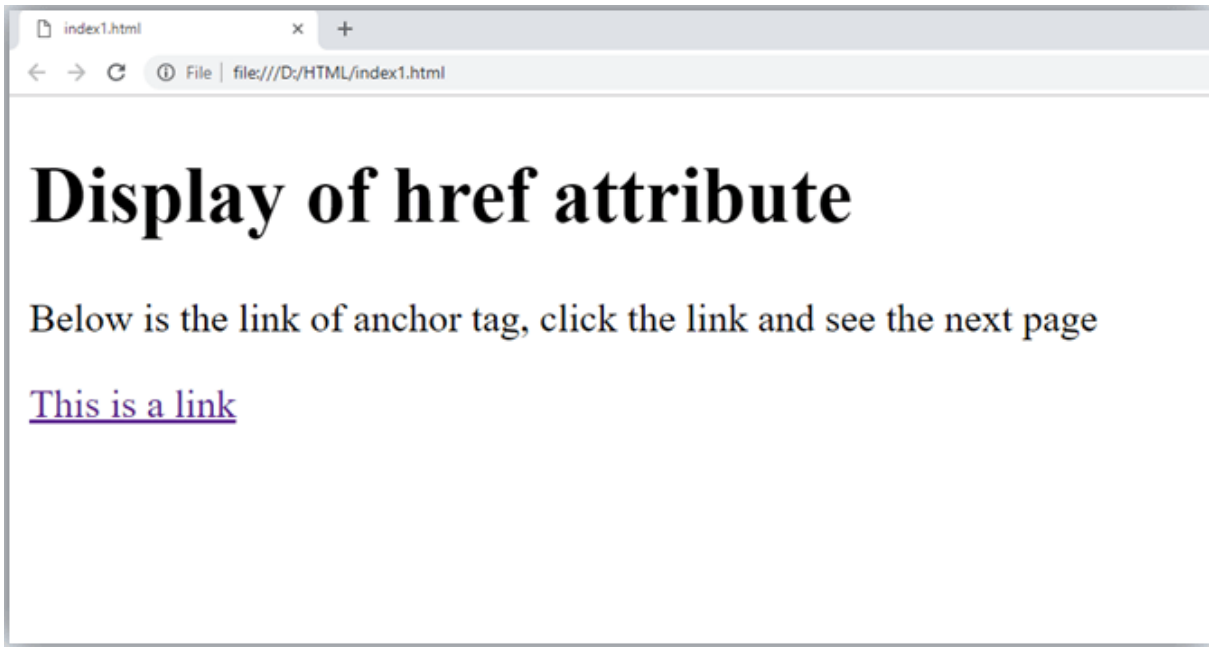
Example

With link address:

```
<a href="https://www.javatpoint.com/html-anchor">This is a link</a>
```

Without link address:

```
<a href="">This is a link</a>
```



The src Attribute

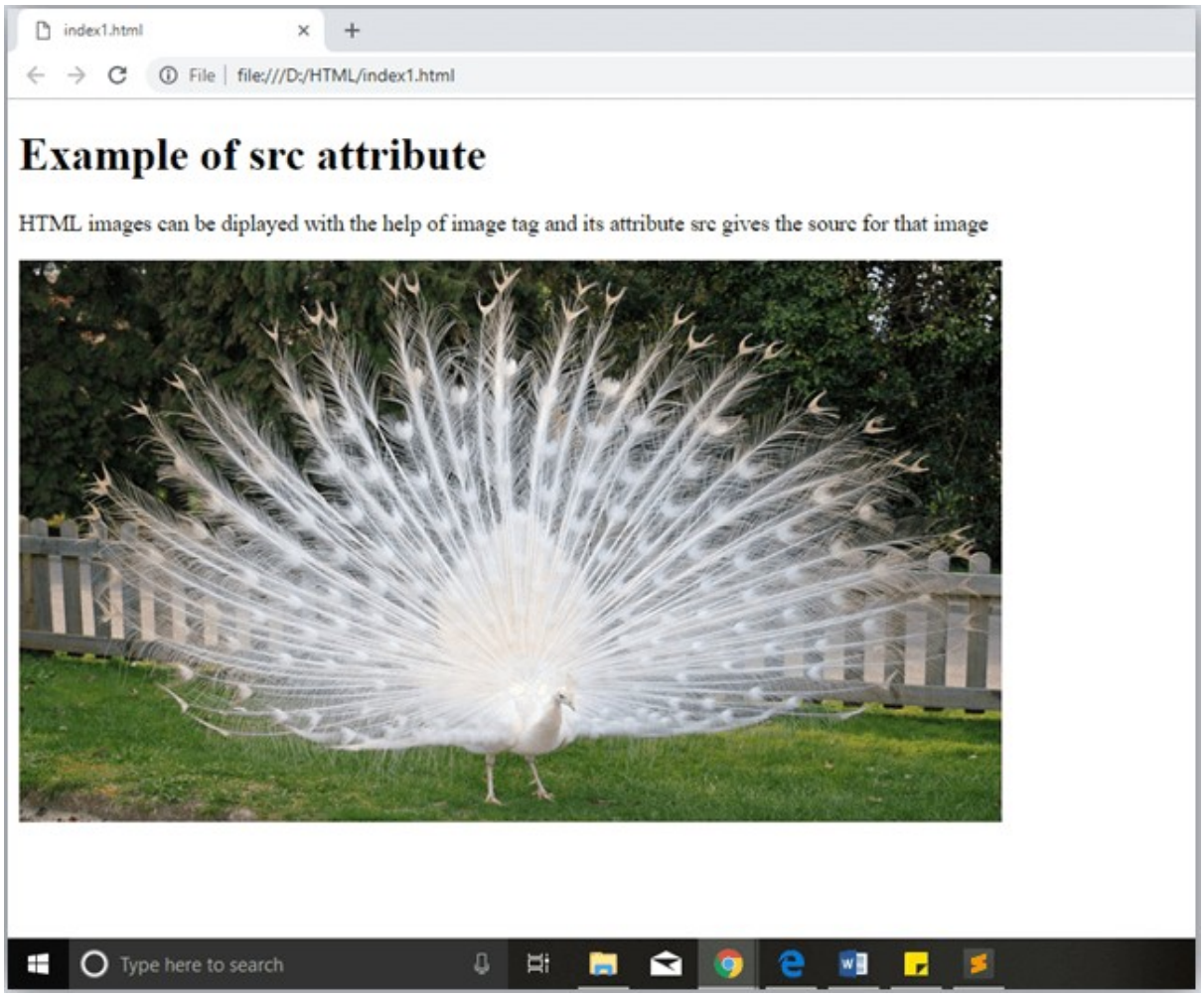
The **src** attribute is one of the important and required attribute of **<img>** element. It is source for the image which is required to display on browser. This attribute can contain image in same directory or another directory. The image name or source should be correct else browser will not display the image.

Example

1. <img src="whitepeacock.jpg" height="400" width="600">

Note: The above example also have height and width attribute, which define the height and width of image on web page.

Output:



Quotes: single quotes or double quotes?

In this chapter you have seen that, we have used attribute with double quotes, but some people might use single quotes in HTML. So use of single quotes with HTML attribute, is also allowed. The following both statements are absolutely fine.

1. A link to HTML.
2. A link to HTML.

IN HTML5, you can also omit use of quotes around attribute values.

1. A link to HTML.

HTML Elements

An HTML file is made of elements. These elements are responsible for creating web pages and define content in that webpage. An element in HTML usually consist of a start tag <tag name>, close tag </tag name> and content inserted between them. **Technically, an element is a collection of start tag, attributes, end tag, content between them**.

Note: Some elements does not have end tag and content, these elements are termed as empty elements or self-closing element or void elements.

Such as:

1. Hello world!!!

Example

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <title>WebPage</title>
5. </head>
6. <body>
7.     <h1>This is my first web page</h1>
8.     <h2> How it looks?</h2>
9.      <p>It looks Nice!!!!!</p>
10. </body>
11. </html>
```

- All the content written between body elements are visible on web page.

Void element: All the elements in HTML do not require to have start tag and end tag, some elements does not have content and end tag such elements are known as Void elements or empty elements. **These elements are also called as unpaired tag**.

**Some Void elements are
 (represents a line break) ,<hr>(represents a horizontal line), etc.**

Nested HTML Elements: HTML can be nested, which means an element can contain another element.

Block-level and Inline HTML elements

For the default display and styling purpose in HTML, all the elements are divided into two categories:

- Block-level element
- Inline element

Block-level element:

- These are the elements, which structure main part of web page, by dividing a page into coherent blocks.
- A block-level element always start with new line and takes the full width of web page, from left to right.
- These elements can contain block-level as well as inline elements.

Following are the block-level elements in HTML.

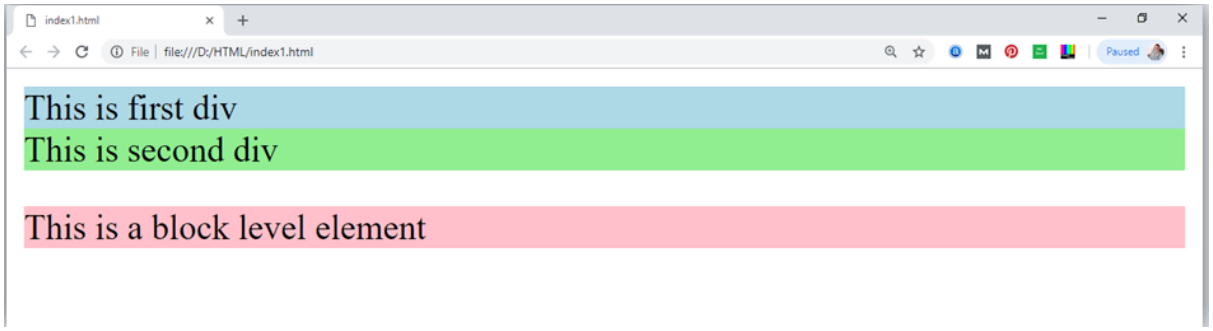
<address>, <article>, <aside>, <blockquote>, <canvas>, <dd>, , <dl>, <dt>, <fieldset>, <figcaption>, <figure>, <footer>, <form>, <h1>-<h6>, <header>, <hr>, <li>, <main>, <nav>, <noscript>, <ol>, <output>, <p>, <pre>, <section>, <table>, <tfoot>, <ul> and <video>.

Note: All these elements are described in later chapters.

Example:

```
<!DOCTYPE html>
<html>
        <head>
  </head>
<body>
   <div style="background-color: lightblue">This is first div</div>
   <div style="background-color: lightgreen">This is second div</div>
   <p style="background-color: pink">This is a block level element</p>
</body>
</html>
```

Output:



In the above example we have used

tag, which defines a section in a web page, and takes full width of page.

We have used style attribute which is used to styling the HTML content, and the background color are showing that it's a block level element.

Inline elements:

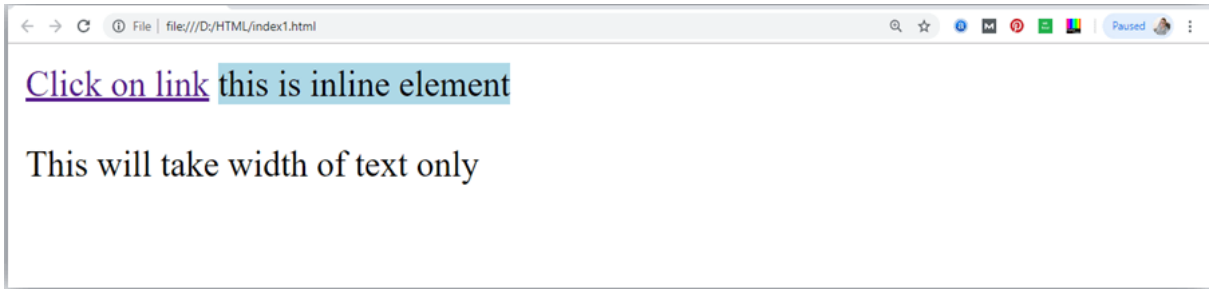
- Inline elements are those elements, which differentiate the part of a given text and provide it a particular function.
- These elements does not start with new line and take width as per requirement.
- The Inline elements are mostly used with other elements.

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

Example:

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.     </head>
5. <body>
6.     <a href="https://www.javatpoint.com/html-tutorial">Click on link</a>
7.     <span style="background-color: lightblue">this is inline element</span>
8.     <p>This will take width of text only</p>
9.  </body>
10. </html>
```

Output:



Following is the list of the some main elements used in HTML:

HTML Formatting

HTML Formatting is a process of formatting text for better look and feel. HTML provides us ability to format text without using CSS. There are many formatting tags in HTML. These tags are used to make text bold, italicized, or underlined. There are almost 14 options available that how text appears in HTML and XHTML.

In HTML the formatting tags are divided into two categories:

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

NOTE: There are some physical and logical tags which may give same visual appearance, but they will be different in semantics.

Here, we are going to learn 14 HTML formatting tags. Following is the list of HTML formatting text.

Element name	Description
<b>	This is a physical tag, which is used to bold the text written between it.
	This is a logical tag, which tells the browser that the text is important.
<i>	This is a physical tag which is used to make text italic.
	This is a logical tag which is used to display content in italic.
<mark>	This tag is used to highlight text.
<u>	This tag is used to underline text written between it.
<tt>	This tag is used to appear a text in teletype. (not supported in HTML5)
<strike>	This tag is used to draw a strikethrough on a section of text. (Not supported in HTML5)
<sup>	It displays the content slightly above the normal line.

<sub>	It displays the content slightly below the normal line.
<del>	This tag is used to display the deleted content.
<ins>	This tag displays the content which is added
<big>	This tag is used to increase the font size by one conventional unit.
<small>	This tag is used to decrease the font size by one unit from base font size.

---

1) Bold Text

HTML and formatting elements

The HTML element is a physical tag which display text in bold font, without any logical importance. If you write anything within ..... element, is shown in bold letters.

See this example:

```
1. <p> <b>Write Your First Paragraph in bold text.</b></p>
```

Output:

Write Your First Paragraph in bold text.

The HTML tag is a logical tag, which displays the content in bold font and informs the browser about its logical importance. If you write anything between ???????., is shown important text.

See this example:

```
1. <p><strong>This is an important content</strong>, and this is normal content</p>
```

Output:

This is an important content, and this is normal content

Example

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <title>formatting elements</title>
5. </head>
6. <body>
7. <h1>Explanation of formatting element</h1>
8. <p><strong>This is an important content</strong>, and this is normal content</p>
9. </body>
10. </html>
```

2) Italic Text

HTML and formatting elements

The HTML element is physical element, which display the enclosed content in italic font, without any added importance. If you write anything within ..... element, is shown in italic letters.

See this example:

```
1. <p> <i>Write Your First Paragraph in italic text.</i></p>
```

Output:

Write Your First Paragraph in italic text.

The HTML tag is a logical element, which will display the enclosed content in italic font, with added semantics importance.

See this example:

```
1. <p><em>This is an important content</em>, which displayed in italic font.</p>
```

Output:

This is an important content, which displayed in italic font.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <title>formatting elements</title>
5. </head>
6. <body>
7. <h1>Explanation of italic formatting element</h1>
8. <p><em>This is an important content</em>, which displayed in italic font.</p>
9. </body>
10. </html>
```

3) HTML Marked formatting

If you want to mark or highlight a text, you should write the content within <mark>.....</mark>.

See this example:

1. <h2> I want to put a <mark> Mark</mark> on your face</h2>

Output:

I want to put a Mark on your face

---

4) Underlined Text

If you write anything within <u>.....</u> element, is shown in underlined text.

See this example:

1. <u>Write Your First Paragraph in underlined text.</u>

Output:

Write Your First Paragraph in underlined text.

---

5) Strike Text

Anything written within <strike>.....</strike> element is displayed with strikethrough. It is a thin line which cross the statement.

See this example:

1. <strike>Write Your First Paragraph with strikethrough</strike>.

Output:

~~Write Your First Paragraph with strikethrough.~~

---

6) Monospaced Font

If you want that each letter has the same width then you should write the content within <tt>.....</tt> element.

Note: We know that most of the fonts are known as variable-width fonts because different letters have different width. (for example: 'w' is wider than 'i'). Monospaced Font provides similar space among every letter.

See this example:

1. Hello <tt>Write Your First Paragraph in monospaced font.</tt>

Output:

Hello Write Your First Paragraph in monospaced font.

---

7) Superscript Text

If you put the content within ^{.....} element, is shown in superscript; means it is displayed half a character's height above the other characters.

See this example:

1. Hello ^{Write Your First Paragraph in superscript.}

Output:

Hello Write Your First Paragraph in superscript.

---

8) Subscript Text

If you put the content within _{.....} element, is shown in subscript ; means it is displayed half a character's height below the other characters.

See this example:

1. Hello _{Write Your First Paragraph in subscript.}

Output:

Hello Write Your First Paragraph in subscript.

HTML Heading

A HTML heading or HTML h tag can be defined as a title or a subtitle which you want to display on the webpage. When you place the text within the heading tags <h1>.....</h1>, it is displayed on the browser in the bold format and size of the text depends on the number of heading.

There are six different HTML headings which are defined with the <h1> to <h6> tags, from highest level h1 (main heading) to the least level h6 (least important heading).

h1 is the largest heading tag and h6 is the smallest one. So h1 is used for most important heading and h6 is used for least important.

Headings in HTML helps the search engine to understand and index the structure of web page.

Note: The main keyword of the whole content of a webpage should be display by h1 heading tag.

See this example:

```
1. <h1>Heading no. 1</h1>
2. <h2>Heading no. 2</h2>
3. <h3>Heading no. 3</h3>
4. <h4>Heading no. 4</h4>
5. <h5>Heading no. 5</h5>
6. <h6>Heading no. 6</h6>
```

Output:

Heading no. 1

Heading no. 2

Heading no. 3

Heading no. 4

Heading no. 5

Heading no. 6

Heading elements (h1....h6) should be used for headings only. They should not be used just to make text bold or big.

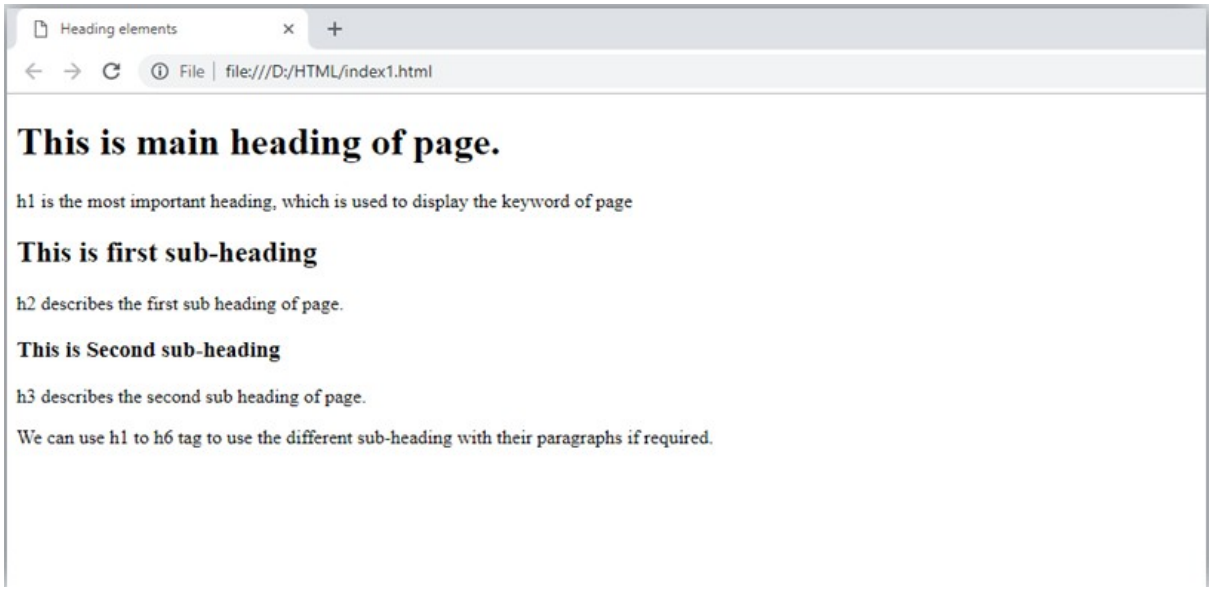
- **HTML headings can also be used with nested elements. Following are different codes to display the way to use heading elements.**

Example:

```
1. <!DOCTYPE html>
2. <html>
3.  <head>
4.    <title>Heading elements</title>
5.  </head>
6.  <body>
7.      <h1>This is main heading of page. </h1>
8.      <p>h1 is the most important heading, which is used to display the keyword of page </p>
9.      <h2>This is first sub-heading</h2>
10.     <p>h2 describes the first sub heading of page. </p>
11.    <h3>This is Second sub-heading</h3>
12.     <p>h3 describes the second sub heading of page.</p>
13.     <p>We can use h1 to h6 tag to use the different sub-
    heading with their paragraphs if
14.                 required.
15.             </p>
16.  </body>
```

17. </html>

Output:



HTML Paragraph

HTML paragraph or HTML p tag is used to define a paragraph in a webpage. Let's take a simple example to see how it work. It is a notable point that a browser itself add an empty line before and after a paragraph. An HTML tag indicates starting of new paragraph.

Note: If we are using various tags in one HTML file then browser automatically adds a single blank line between the two paragraphs.

See this example:

1. This is first paragraph.
2. This is second paragraph.
3. This is third paragraph.

Output:

This is first paragraph.

This is second paragraph.

This is third paragraph.

Space inside HTML Paragraph

If you put a lot of spaces inside the HTML p tag, browser removes extra spaces and extra line while displaying the page. The browser counts number of spaces and lines as a single one.

```
<p>
I am
going to provide
you a tutorial on HTML
and hope that it will
be very beneficial for you.
</p>
<p>
Look, I put here a lot
of spaces                but          I know, Browser will ignore it.
</p>
<p>
You cannot determine the display of HTML</p>
<p>because resized windows may create different result.
</p>
```

Output:

I am going to provide you a tutorial on HTML and hope that it will be very beneficial for you.

Look, I put here a lot of spaces but I know, Browser will ignore it.

You cannot determine the display of HTML

because resized windows may create different result.

As you can see, all the extra lines and unnecessary spaces are removed by the browser.

How to Use
 and <hr> tag with paragraph?

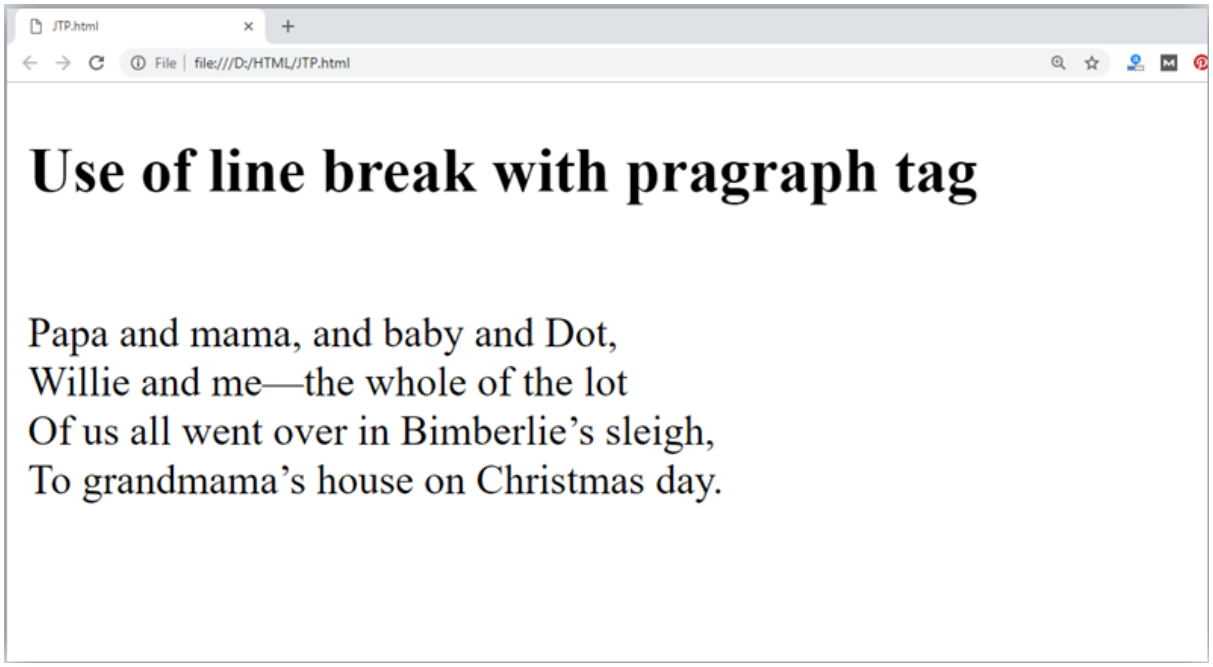
An HTML
 tag is used for line break and it can be used with paragraph elements. Following is the example to show how to use
 with <p> element.

Example:

```
<!DOCTYPE html>
<html>
    <head>
   </head>
  <body>
     <h2> Use of line break with pragraph tag</h2>
        <p><br>Papa and mama, and baby and Dot,
    <br>Willie and me?the whole of the lot
          <br>Of us all went over in Bimberlie's sleigh,
```

```
10.             <br>To grandmama's house on Christmas day.
11.         </p>
12.   </body>
13. </html>
```

Output:

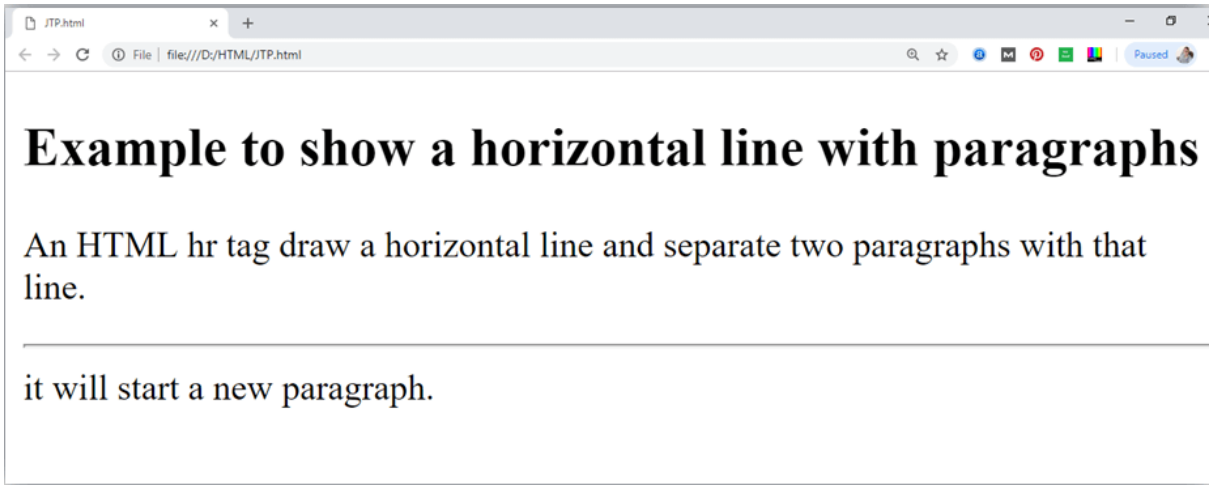


An HTML <hr> tag is used to apply a horizontal line between two statements or two paragraphs. Following is the example which is showing use of <hr> tag with paragraph.

Example:

```
1.  <!DOCTYPE html>
2.  <html>
3.   <head>
4.     </head>
5.   <body>
6.     <h2> Example to show a horizontal line with paragraphs</h2>
7.       <p> An HTML hr tag draw a horizontal line and separate two paragraphs with that
     line.<hr> it will start a new paragraph.
8.       </p>
9.    </body>
10. </html>
```

Output:



HTML Anchor

The **HTML anchor tag** defines *a hyperlink that links one page to another page*. It can create hyperlink to other web page as well as files, location, or any URL. The "href" attribute is the most important attribute of the HTML a tag. and which links to destination page or URL.

href attribute of HTML anchor tag

The href attribute is used to define the address of the file to be linked. In other words, it points out the destination page.

The syntax of HTML anchor tag is given below.

 Link Text

Let's see an example of HTML anchor tag.

1. Click for Second Page

Specify a location for Link using target attribute

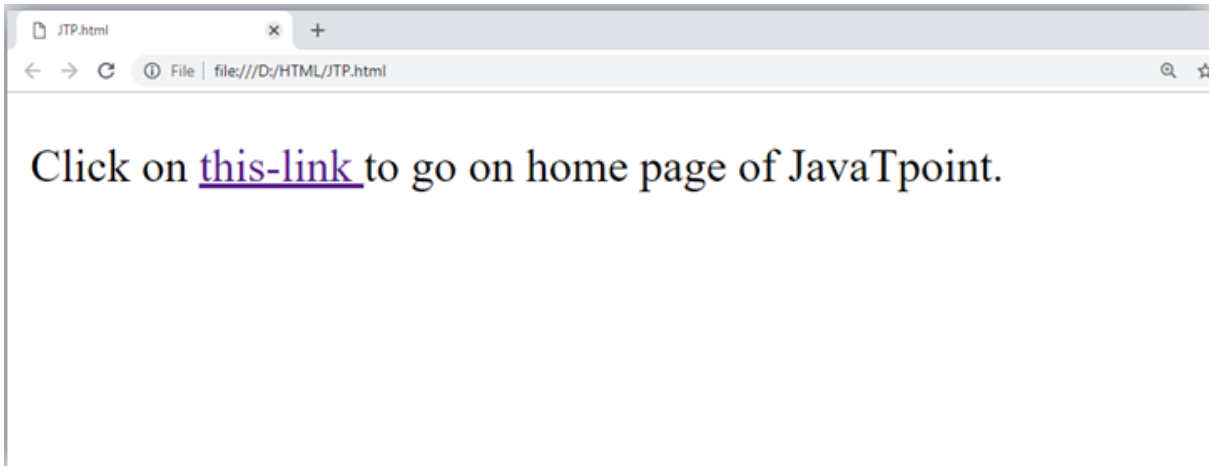
If we want to open that link to another page then we can use target attribute of <a> tag. With the help of this link will be open in next page.

Example:

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title></title>

```
5. </head>
6. <body>
7. <p>Click on <a href="https://www.javatpoint.com/" target="_blank"> this-
   link </a>to go on home page of JavaTpoint.</p>
8. </body>
9. </html>
```

Output:



Note:

- The **target** attribute can only use with href attribute in anchor tag.
- If we will not use target attribute then link will open in same page.

Appearance of HTML anchor tag

An **unvisited link** is displayed underlined and blue.

A **visited link** displayed underlined and purple.

An **active link** is underlined and red.

HTML Image

HTML img tag is used to display image on the web page. HTML img tag is an empty tag that contains attributes only, closing tags are not used in HTML image element.

Let's see an example of HTML image.

```
1. <h2>HTML Image Example</h2>
2. <img src="good_morning.jpg" alt="Good Morning Friends"/>
```

Output:

Attributes of HTML img tag

The src and alt are important attributes of HTML img tag. All attributes of HTML image tag are given below.

1) src

It is a necessary attribute that describes the source or path of the image. It instructs the browser where to look for the image on the server.

The location of image may be on the same directory or another server.

2) alt

The alt attribute defines an alternate text for the image, if it can't be displayed. The value of the alt attribute describe the image in words. The alt attribute is considered good for SEO prospective.

3) width

It is an optional attribute which is used to specify the width to display the image. It is not recommended now. You should apply CSS in place of width attribute.

4) height

It h3 the height of the image. The HTML height attribute also supports iframe, image and object elements. It is not recommended now. You should apply CSS in place of height attribute.

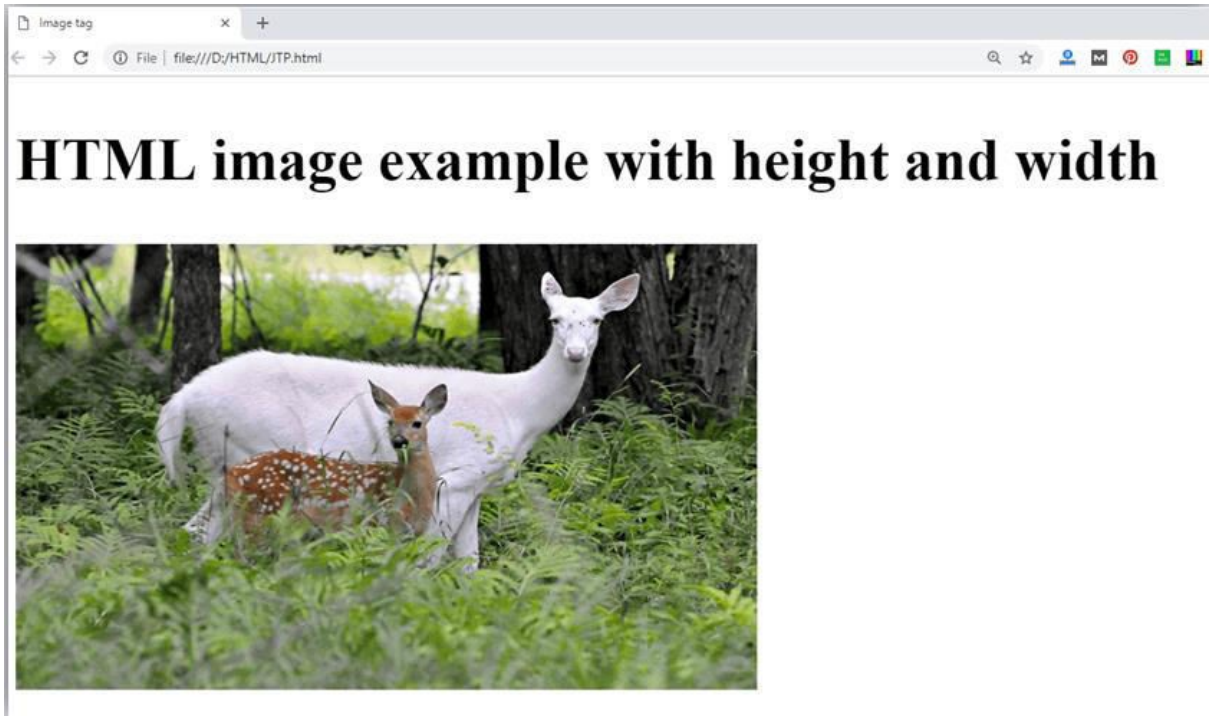
Use of height and width attribute with img tag

You have learnt about how to insert an image in your web page, now if we want to give some height and width to display image according to our requirement, then we can set it with height and width attributes of image.

Example:

1. <img src="animal.jpg" height="180" width="300" alt="animal image">

Output:



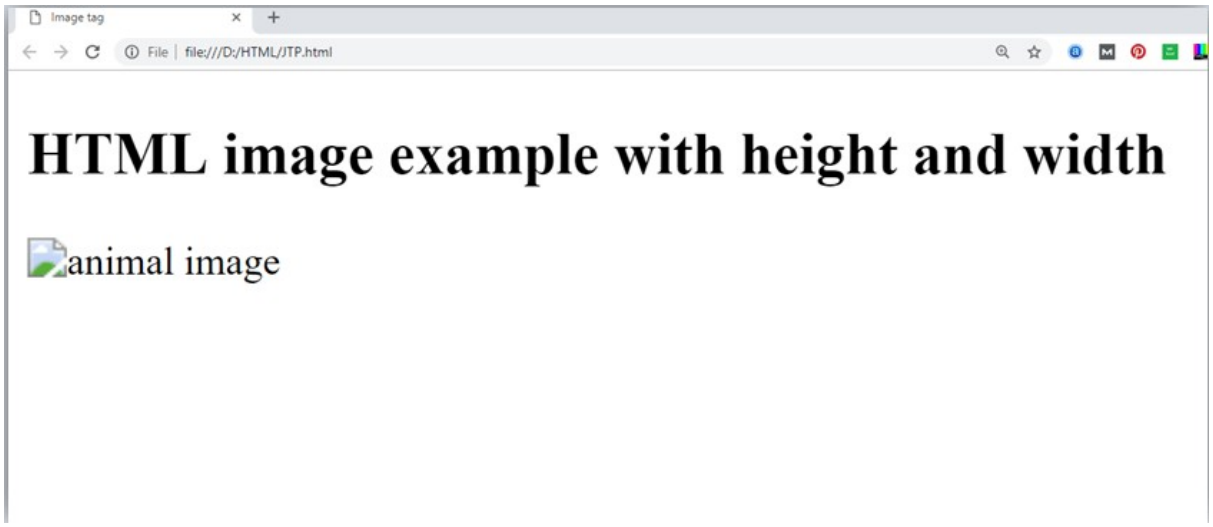
Note: Always try to insert the image with height and width, else it may flicker while displaying on webpage.

Use of alt attribute

We can use alt attribute with tag. It will display an alternative text in case if image cannot be displayed on browser. Following is the example for alt attribute:

1. <img src="animal.png" height="180" width="300" alt="animal image">

Output:



How to get image from another directory/folder?

To insert an image in your web, that image must be present in your same folder where you have put the HTML file. But if in some case image is available in some other directory then you can access the image like this:

1. <img src="E:/images/animal.png" height="180" width="300" alt="animal image">

In above statement we have put image in local disk E------>images folder------>animal.png.

Note: If src URL will be incorrect or misspell then it will not display your image on web page, so try to put correct URL.

Use <img> tag as a link

We can also link an image with other page or we can use an image as a link. To do this, put <img> tag inside the <a> tag.

Example:

1. <img src="robot.jpg" height="100" width="100">

Output:

HTML Table

HTML table tag is used to display data in tabular form (row * column). There can be many columns in a row.

We can create a table to display data in tabular form, using <table> element, with the help of <tr> ,<td>, and <th> elements.

In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags.

HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use div tag over table to manage the layout of the page .

HTML Table Tags

Tag	Description
<table>	It defines a table.
<tr>	It defines a row in a table.
<th>	It defines a header cell in a table.
<td>	It defines a cell in a table.
<caption>	It defines the table caption.
<colgroup>	It specifies a group of one or more columns in a table for formatting.
<col>	It is used with <colgroup> element to specify column properties for each column.
<tbody>	It is used to group the body content in a table.
<thead>	It is used to group the header content in a table.
<tfooter>	It is used to group the footer content in a table.

HTML Table Example

Let's see the example of HTML table tag. It output is shown above.

```
1. <table>
2. <tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
3. <tr><td>Sonoo</td><td>Jaiswal</td><td>60</td></tr>
4. <tr><td>James</td><td>William</td><td>80</td></tr>
5. <tr><td>Swati</td><td>Sironi</td><td>82</td></tr>
6. <tr><td>Chetna</td><td>Singh</td><td>72</td></tr>
7. </table>
```

Output:

First_Name	Last_Name	Marks
Sonoo	Jaiswal	60

HTML 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:

1. Ordered List or Numbered List (ol)
2. Unordered List or Bulleted List (ul)
3. Description List or Definition List (dl)

Note: We can create a list inside another list, which will be termed as nested List.

HTML Ordered List or Numbered List

In the ordered HTML lists, all the list items are marked with numbers by default. It is known as numbered list also. The ordered list starts with <ol> tag and the list items start with <li> tag.

```
1. <ol>
2.  <li>Aries</li>
3.  <li>Bingo</li>
4.  <li>Leo</li>
5.  <li>Oracle</li>
6. </ol>
```

Output:

1. Aries
2. Bingo
3. Leo
4. Oracle

Click here for full details of HTML ordered list. HTML Ordered List

HTML Unordered List or Bulleted List

In HTML Unordered list, all the list items are marked with bullets. It is also known as bulleted list also. The Unordered list starts with <ul> tag and list items start with the <li> tag.

```
<ul>
 <li>Aries</li>
 <li>Bingo</li>
 <li>Leo</li>
 <li>Oracle</li>
</ul>
```

Output:

- Aries
- Bingo
- Leo
- Oracle

HTML Description List or Definition List

HTML Description list is also a list style which is supported by HTML and XHTML. It is also known as definition list where entries are listed like a dictionary or encyclopedia.

The definition list is very appropriate when you want to present glossary, list of terms or other name-value list.

The HTML definition list contains following three tags:

1. **<dl> tag** defines the start of the list.
2. **<dt> tag** defines a term.
3. **<dd> tag** defines the term definition (description).

```
<dl>
  <dt>Aries</dt>
  <dd>-One of the 12 horoscope sign.</dd>
  <dt>Bingo</dt>
  <dd>-One of my evening snacks</dd>
 <dt>Leo</dt>
 <dd>-It is also an one of the 12 horoscope sign.</dd>
  <dt>Oracle</dt>
```

```
    <dd>-It is a multinational technology corporation.</dd>
</dl>
```

Output:

Aries

-One of the 12 horoscope sign.

Bingo

-One of my evening snacks

Leo

-It is also an one of the 12 horoscope sign.

Oracle

-It is a multinational technology corporation.

Click here for full details of HTML description list. HTML Description List

---

HTML Nested List

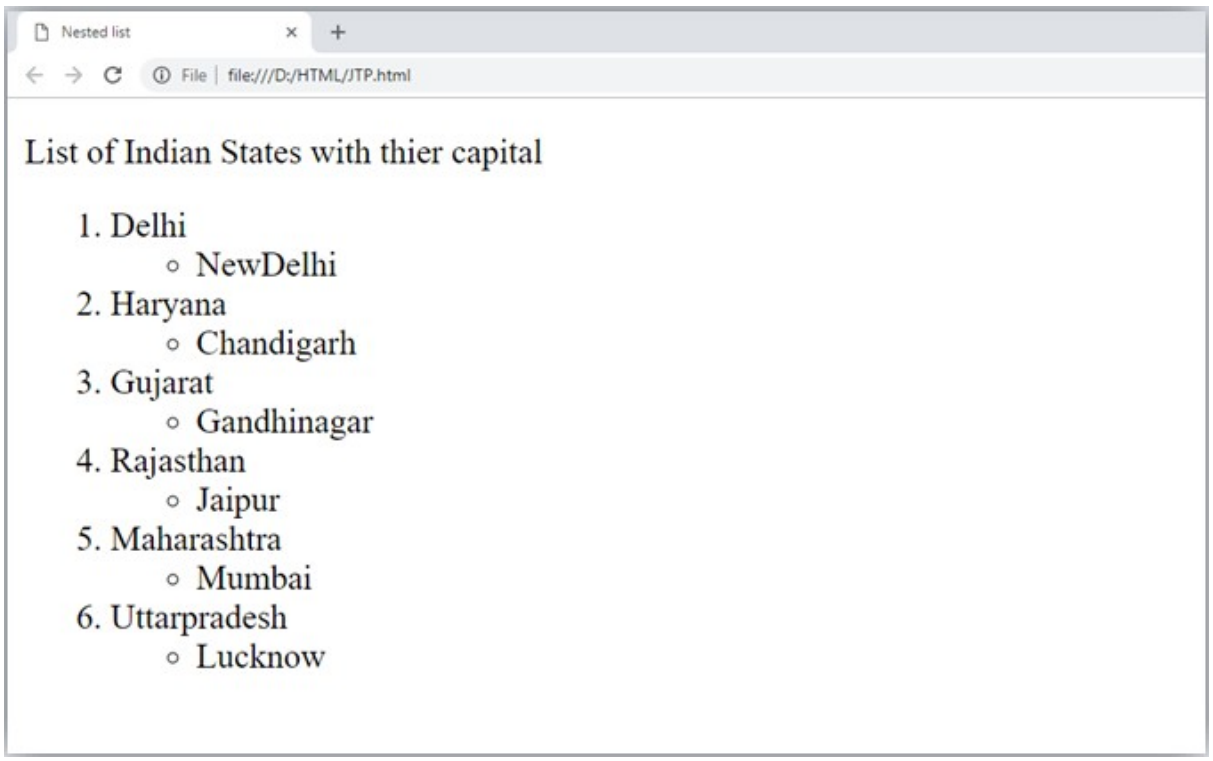
A list within another list is termed as nested list. If you want a bullet list inside a numbered list then such type of list will called as nested list.

Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Nested list</title>
</head>
<body>
   <p>List of Indian States with thier capital</p>
<ol>
   <li>Delhi
      <ul>
         <li>NewDelhi</li>
      </ul>
   </li>
   <li>Haryana
      <ul>
         <li>Chandigarh</li>
      </ul>
   </li>
   <li>Gujarat
      <ul>
```

```
21.             <li>Gandhinagar</li>
22.         </ul>
23.     </li>
24.     <li>Rajasthan
25.         <ul>
26.             <li>Jaipur</li>
27.         </ul>
28.     </li>
29.     <li>Maharashtra
30.         <ul>
31.             <li>Mumbai</li>
32.         </ul>
33.     </li>
34.     <li>Uttarpradesh
35.         <ul>
36.             <li>Lucknow</li></ul>
37.     </li>
38. </ol>
39. </body>
40. </html>
```

Output:



HTML Ordered List | HTML Numbered List

HTML Ordered List or Numbered List displays elements in numbered format. The HTML ol tag is used for ordered list. We can use ordered list to represent items either in numerical order format or alphabetical order format, or any format where an order is emphasized. There can be different types of numbered list:

- Numeric Number (1, 2, 3)
- Capital Roman Number (I II III)
- Small Romal Number (i ii iii)
- Capital Alphabet (A B C)
- Small Alphabet (a b c)

To represent different ordered lists, there are 5 types of attributes in <ol> tag.

Type	Description
Type "1"	This is the default type. In this type, the list items are numbered with numbers.
Type "I"	In this type, the list items are numbered with upper case roman numbers.
Type "i"	In this type, the list items are numbered with lower case roman numbers.
Type "A"	In this type, the list items are numbered with upper case letters.
Type "a"	In this type, the list items are numbered with lower case letters.

HTML Ordered List Example

Let's see the example of HTML ordered list that displays 4 topics in numbered list. Here we are not defining type="1" because it is the default type.

```
1. <ol>
2.  <li>HTML</li>
3.  <li>Java</li>
4.  <li>JavaScript</li>
5.  <li>SQL</li>
6. </ol>
```

Output:

1. HTML
2. Java
3. JavaScript
4. SQL

---

ol type="I"

Let's see the example to display list in roman number uppercase.

```
1. <ol type="I">
```

```
2.  <li>HTML</li>
3.  <li>Java</li>
4.  <li>JavaScript</li>
5.  <li>SQL</li>
6. </ol>
```

Output:

I. HTML
II. Java
III. JavaScript
IV. SQL

---

ol type="i"

Let's see the example to display list in roman number lowercase.

```
1. <ol type="i">
2.  <li>HTML</li>
3.  <li>Java</li>
4.  <li>JavaScript</li>
5.  <li>SQL</li>
6. </ol>
```

Output:

i. HTML
ii. Java
iii. JavaScript
iv. SQL

---

ol type="A"

Let's see the example to display list in alphabet uppercase.

```
1. <ol type="A">
2.  <li>HTML</li>
3.  <li>Java</li>
4.  <li>JavaScript</li>
5.  <li>SQL</li>
6. </ol>
```

Output:

A. HTML

B. Java
C. JavaScript
D. SQL

---

ol type="a"

Let's see the example to display list in alphabet lowercase.

```
<ol type="a">
 <li>HTML</li>
 <li>Java</li>
 <li>JavaScript</li>
 <li>SQL</li>
</ol>
```

Output:

a. HTML
b. Java
c. JavaScript
d. SQL

---

start attribute

The start attribute is used with ol tag to specify from where to start the list items.

<ol type="1" start="5"> : It will show numeric values starting with "5".

<ol type="A" start="5"> : It will show capital alphabets starting with "E".

<ol type="a" start="5"> : It will show lower case alphabets starting with "e".

<ol type="I" start="5"> : It will show Roman upper case value starting with "V".

<ol type="i" start="5"> : It will show Roman lower case value starting with "v".

```
<ol type="i" start="5">
 <li>HTML</li>
 <li>Java</li>
 <li>JavaScript</li>
 <li>SQL</li>
</ol>
```

Output:

v. HTML
vi. Java
vii. JavaScript
viii. SQL

HTML Form

An **HTML form** is *a section of a document* which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

An HTML form facilitates the user to enter data that is to be sent to the server for processing such as name, email address, password, phone number, etc. .

Why use HTML Form

HTML forms are required if you want to collect some data from of the site visitor.

For example: If a user want to purchase some items on internet, he/she must fill the form such as shipping address and credit/debit card details so that item can be sent to the given address.

HTML Form Syntax

```
1. <form action="server url" method="get|post">
2.   //input controls e.g. textfield, textarea, radiobutton, button
3. </form>
```

HTML Form Tags

Let's see the list of HTML 5 form tags.

Tag	Description
<form>	It defines an HTML form to enter inputs by the used side.
<input>	It defines an input control.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element.
<fieldset>	It groups the related element in a form.

<legend>	It defines a caption for a <fieldset> element.
<select>	It defines a drop-down list.
<optgroup>	It defines a group of related options in a drop-down list.
<option>	It defines an option in a drop-down list.
<button>	It defines a clickable button.

HTML 5 Form Tags

Let's see the list of HTML 5 form tags.

Tag	Description
<datalist>	It specifies a list of pre-defined options for input control.
<keygen>	It defines a key-pair generator field for forms.
<output>	It defines the result of a calculation.

HTML <form> element

The HTML <form> element provide a document section to take input from user. It provides various interactive controls for submitting information to web server such as text field, text area, password field, etc.

Note: The <form> element does not itself create a form but it is container to contain all required form elements, such as <input>, <label>, etc.

Syntax:

```
1. <form>
2. //Form elements
3. </form>
```

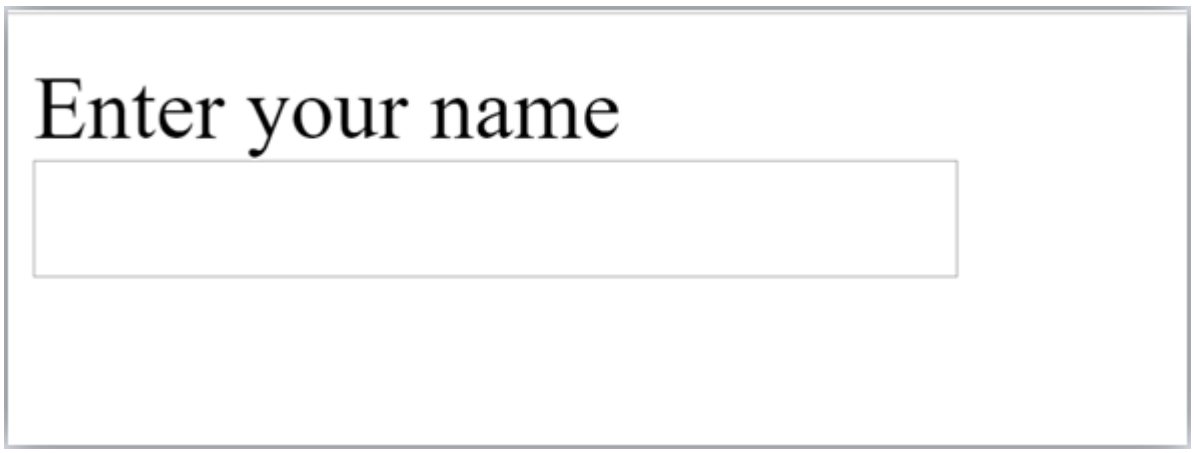
HTML <input> element

The HTML <input> element is fundamental form element. It is used to create form fields, to take input from user. We can apply different input filed to gather different information form user. Following is the example to show the simple text input.

Example:

```
<body>
  <form>
    Enter your name  <br>
    <input type="text" name="username">
  </form>
</body>
```

Output:

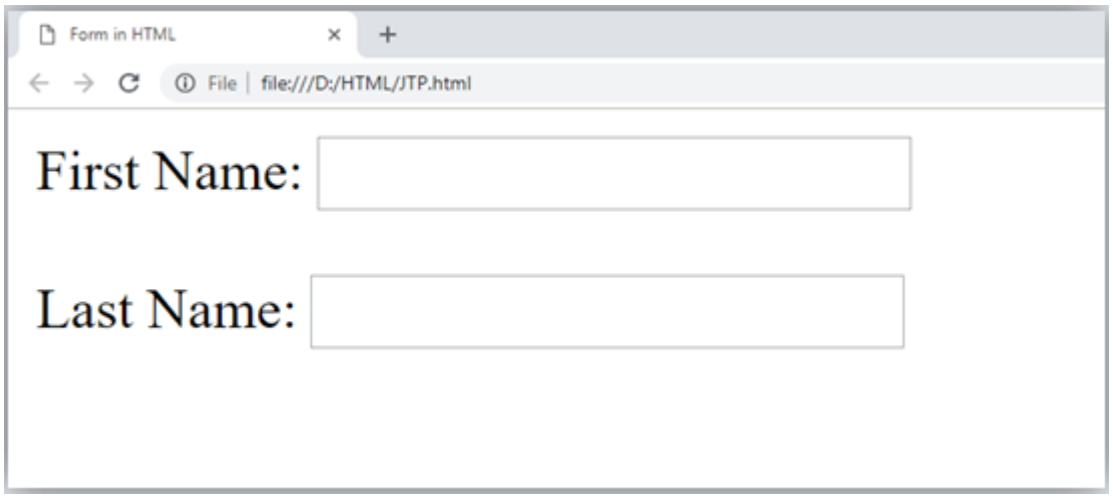


HTML TextField Control

The type="text" attribute of input tag creates textfield control also known as single line textfield control. The name attribute is optional, but it is required for the server side component such as JSP, ASP, PHP etc.

```
<form>
    First Name: <input type="text" name="firstname"/> <br/>
    Last Name:  <input type="text" name="lastname"/> <br/>
</form>
```

Output:



Note: If you will omit 'name' attribute then the text filed input will not be submitted to server.

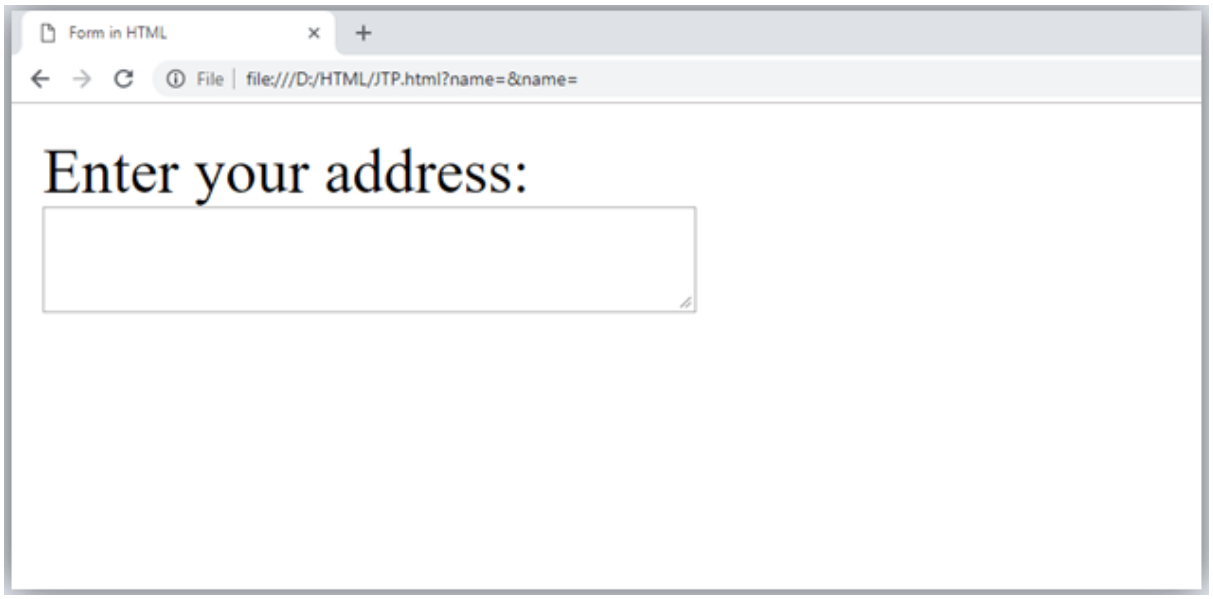
HTML <textarea> tag in form

The <textarea> tag in HTML is used to insert multiple-line text in a form. The size of <textarea> can be specify either using "rows" or "cols" attribute or by CSS.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Form in HTML</title>
</head>
<body>
  <form>
       Enter your address:<br>
      <textarea rows="2" cols="20"></textarea>
  </form>
</body>
</html>
```

Output:



Label Tag in Form

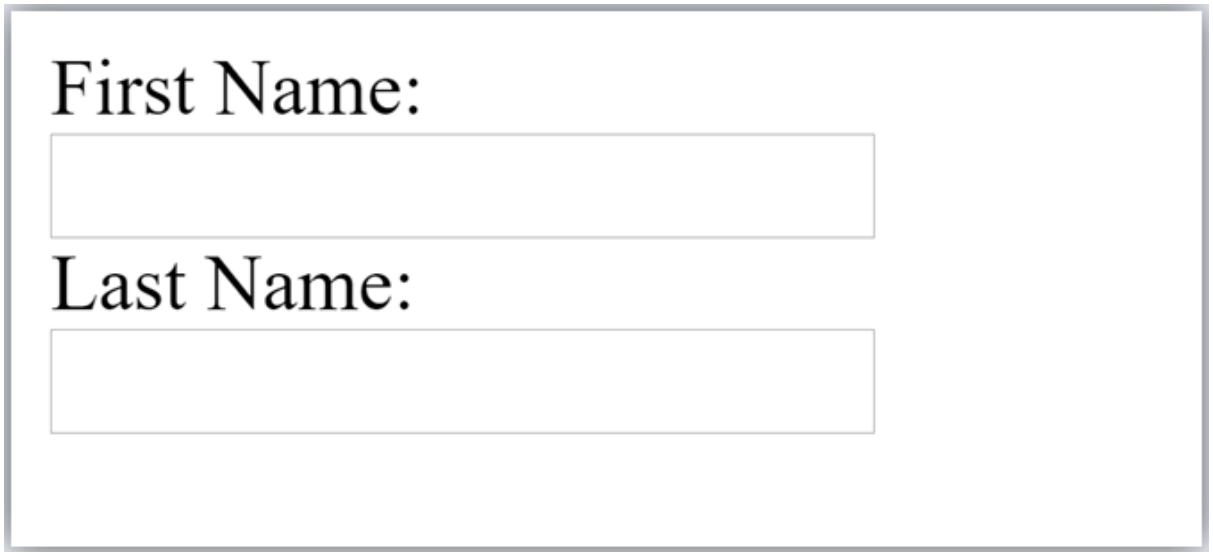
It is considered better to have label in form. As it makes the code parser/browser/user friendly.

If you click on the label tag, it will focus on the text control. To do so, you need to have for attribute in label tag that must be same as id attribute of input tag.

NOTE: It is good to use <label> tag with form, although it is optional but if you will use it, then it will provide a focus when you tap or click on label tag. It is more worthy with touchscreens.

```
<form>
   <label for="firstname">First Name: </label> <br/>
            <input type="text" id="firstname" name="firstname"/> <br/>
   <label for="lastname">Last Name: </label>
            <input type="text" id="lastname" name="lastname"/> <br/>
 </form>
```

Output:



HTML Password Field Control

The password is not visible to the user in password field control.

```
<form>
   <label for="password">Password: </label>
            <input type="password" id="password" name="password"/> <br/>
</form>
```

Output:

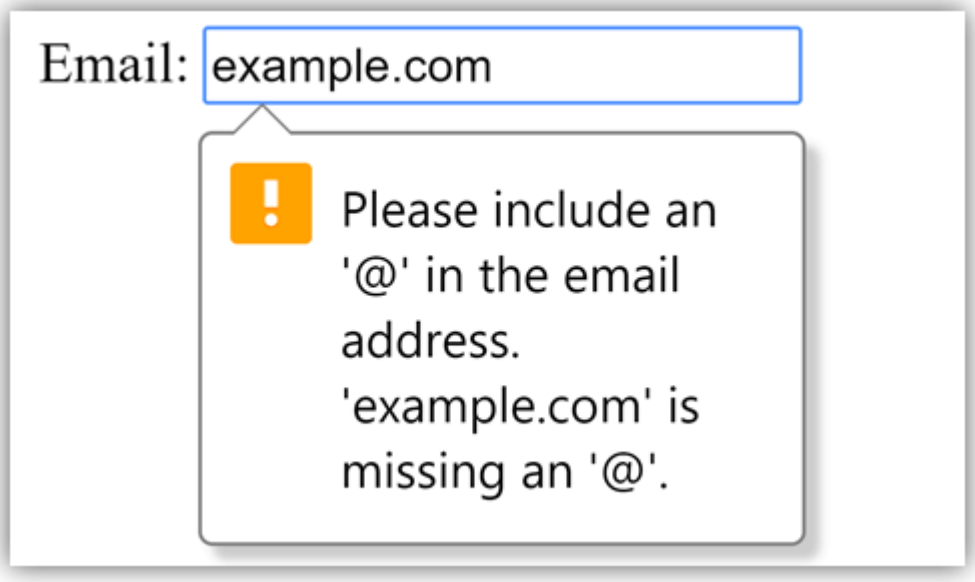
HTML 5 Email Field Control

The email field in new in HTML 5. It validates the text for correct email address. You must use @ and . in this field.

```
1. <form>
2.    <label for="email">Email: </label>
3.          <input type="email" id="email" name="email"/> <br/>
4. </form>
```

It will display in browser like below:

Note: If we will not enter the correct email, it will display error like:



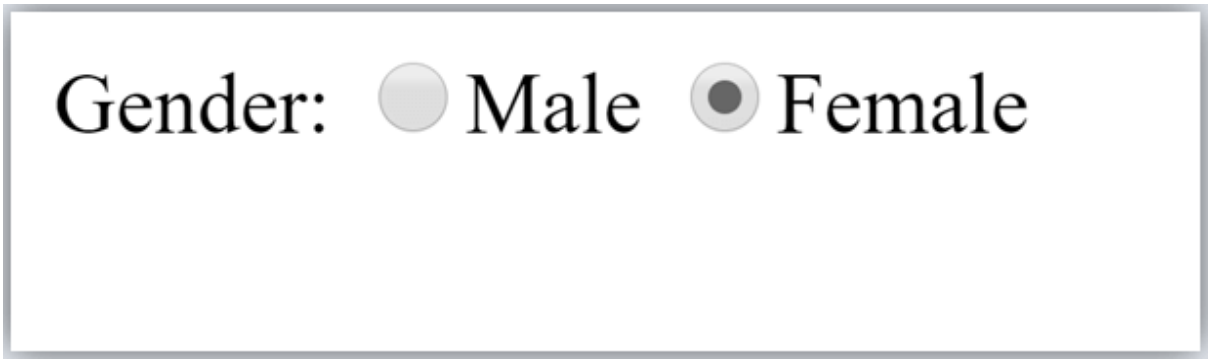
Radio Button Control

The radio button is used to select one option from multiple options. It is used for selection of gender, quiz questions etc.

If you use one name for all the radio buttons, only one radio button can be selected at a time.

Using radio buttons for multiple options, you can only choose a single option at a time.

```
<form>
   <label for="gender">Gender: </label>
        <input type="radio" id="gender" name="gender" value="male"/>Male
       <input type="radio" id="gender" name="gender" value="female"/>Female <br/>
</form>
```



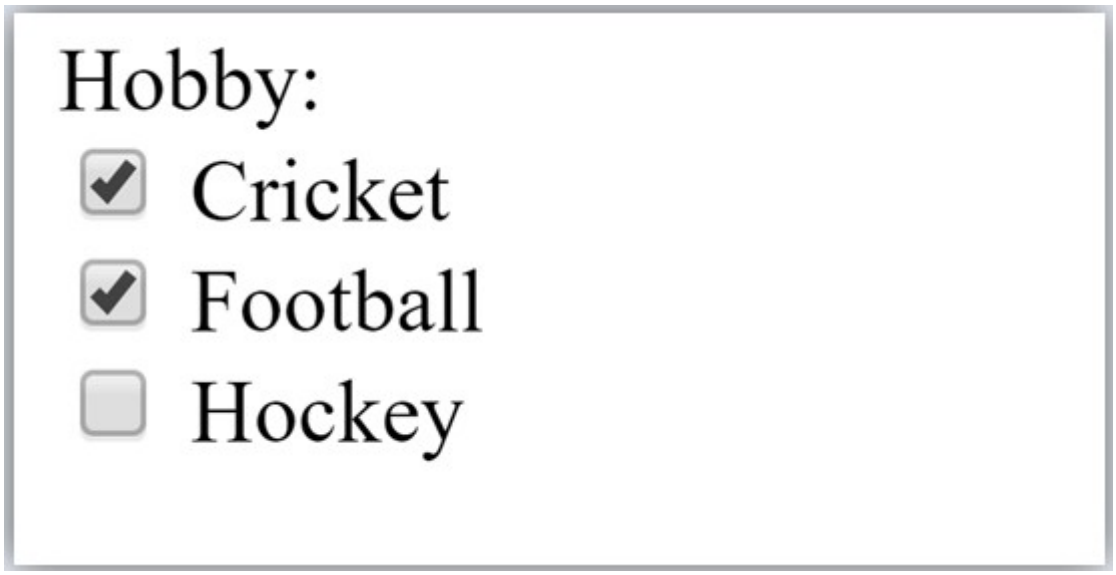
Checkbox Control

The checkbox control is used to check multiple options from given checkboxes.

```
<form>
Hobby:<br>
        <input type="checkbox" id="cricket" name="cricket" value="cricket"/>
         <label for="cricket">Cricket</label> <br>
        <input type="checkbox" id="football" name="football" value="football"/>
         <label for="football">Football</label> <br>
        <input type="checkbox" id="hockey" name="hockey" value="hockey"/>
         <label for="hockey">Hockey</label>
</form>
```

Note: These are similar to radio button except it can choose multiple options at a time and radio button can select one button at a time, and its display.

Output:



Submit button control

HTML **<input type="submit">** are used to add a submit button on web page. When user clicks on submit button, then form get submit to the server.

Syntax:

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

The type = submit , specifying that it is a submit button

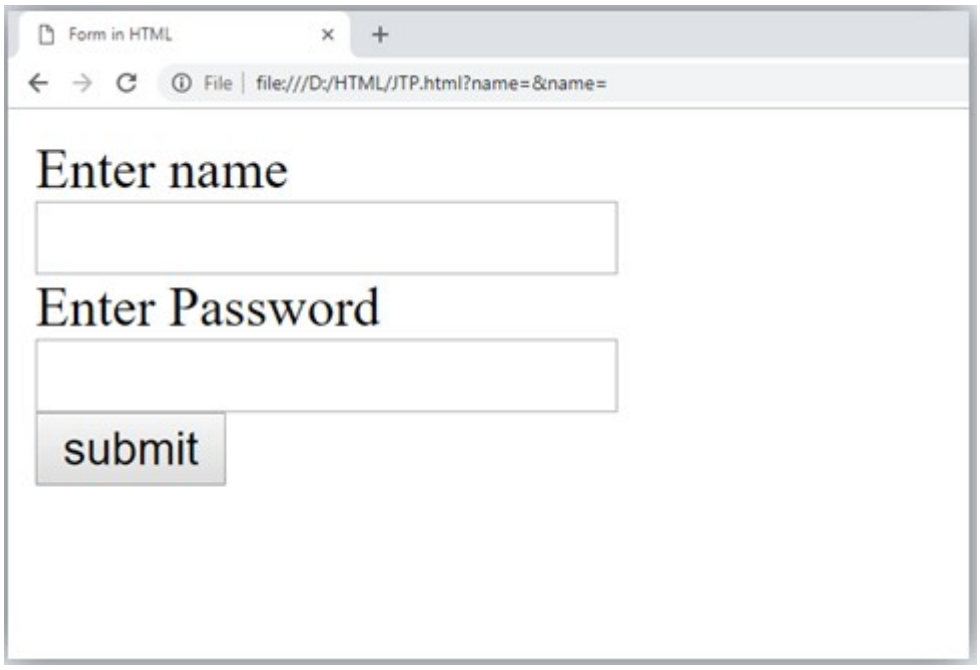
The value attribute can be anything which we write on button on web page.

The name attribute can be omit here.

Example:

```
<form>
    <label for="name">Enter name</label><br>
    <input type="text" id="name" name="name"><br>
    <label for="pass">Enter Password</label><br>
    <input type="Password" id="pass" name="pass"><br>
    <input type="submit" value="submit">
</form>
```

Output:



HTML <fieldset> element:

The <fieldset> element in HTML is used to group the related information of a form. This element is used with <legend> element which provide caption for the grouped elements.

Example:

```
1.  <form>
2.      <fieldset>
3.       <legend>User Information:</legend>
4.     <label for="name">Enter name</label><br>
5.  <input type="text" id="name" name="name"><br>
6.  <label for="pass">Enter Password</label><br>
7.  <input type="Password" id="pass" name="pass"><br>
8.  <input type="submit" value="submit">
9.  </fieldset>
10. lt;/form>
```

Output:

HTML Form Example

Following is the example for a simple form of registration.

```
<!DOCTYPE html>
<html>
<head>
<title>Form in HTML</title>
</head>
<body>
<h2>Registration form</h2>
<form>
<fieldset>
<legend>User personal information</legend>
<label>Enter your full name</label><br>
<input type="text" name="name"><br>
<label>Enter your email</label><br>
<input type="email" name="email"><br>
<label>Enter your password</label><br>
<input type="password" name="pass"><br>
<label>confirm your password</label><br>
<input type="password" name="pass"><br>
<br><label>Enter your gender</label><br>
<input type="radio" id="gender" name="gender" value="male"/>Male <br>
<input type="radio" id="gender" name="gender" value="female"/>Female <br/
>
<input type="radio" id="gender" name="gender" value="others"/>others <br/>
<br>Enter your Address:<br>
<textarea></textarea><br>
<input type="submit" value="sign-up">
</fieldset>
</form>
</body>
</html>
```

Output:

HTML Form Input Types

In HTML <input type=" "> is an important element of HTML form. The "type" attribute of input element can be various types, which defines information field. Such as <input type="text" name="name"> gives a text box.

Following is a list of all types of <input> element of HTML.

type=" "	Description
text	Defines a one-line text input field
password	Defines a one-line password input field
submit	Defines a submit button to submit the form to server
reset	Defines a reset button to reset all values in the form.
radio	Defines a radio button which allows select one option.
checkbox	Defines checkboxes which allow select multiple options form.
button	Defines a simple push button, which can be programmed to perform a task on an event.
file	Defines to select the file from device storage.
image	Defines a graphical submit button.

HTML5 added new types on <input> element. Following is the list of types of elements of HTML5

type=" "	Description
color	Defines an input field with a specific color.
date	Defines an input field for selection of date.
datetime-local	Defines an input field for entering a date without time zone.
email	Defines an input field for entering an email address.
month	Defines a control with month and year, without time zone.
number	Defines an input field to enter a number.
url	Defines a field for entering URL
week	Defines a field to enter the date with week-year, without time zone.
search	Defines a single line text field for entering a search string.
tel	Defines an input field for entering the telephone number.

Following is the description about types of <input> element with examples.

1. <input type="text">:

<input> element of type "text" are used to define a single-line input text field.

Example:

```
1. <form>
2.     <label>Enter first name</label><br>
3.     <input type="text" name="firstname"><br>
4.     <label>Enter last name</label><br>
5.     <input type="text" name="lastname"><br>
6.     <p><strong>Note:</strong>The default maximum cahracter lenght is 20.</p>
7. </form>
```

Output:

Input "text" type:

The **"text"** field defines a sinlge line input text field.

Enter first name

Enter last name

Note: The default maximum cahracterlenght is 20.

---

2. <input type="password">:

The <input> element of type "password" allow a user to enter the password securely in a webpage. The entered text in password filed converted into "*" or ".", so that it cannot be read by another user.

Example:

```
<form>
   <label>Enter User name</label><br>
   <input type="text" name="firstname"><br>
   <label>Enter Password</label><br>
   <input type="Password" name="password"><br>
   <br><input type="submit" value="submit">
</form>
```

Output:

Input "password" type:

The **"password"** field defines a sinlge line input password field to enter the password securely.

Enter User name

Enter Password

---

3. <input type="submit">:

The <input> element of type "submit" defines a submit button to submit the form to the server when the "click" event occurs.

Example:

```
<form action="https://www.javatpoint.com/html-tutorial">
   <label>Enter User name</label><br>
   <input type="text" name="firstname"><br>
   <label>Enter Password</label><br>
   <input type="Password" name="password"><br>
   <br><input type="submit" value="submit">
```

```
7. </form>
```

Output:

Input "submit" type:
Enter User name

Enter Password

After clicking on submit button, this will submit the form to server and will redirect the page to **action** value.We will learn about "action" attribute in later chapters

---

4. <input type="reset">:

The <input> type "reset" is also defined as a button but when the user performs a click event, it by default reset the all inputted values.

Example:

```
1. <form>
2.    <label>User id: </label>
3.     <input type="text" name="user-id" value="user">
4.           <label>Password: </label>
5.     <input type="password" name="pass" value="pass"><br><br>
6.     <input type="submit" value="login">
7.      <input type="reset" value="Reset">
8. </form>
```

Output:

Input "reset" type:
User id: user Password: ****

Try to change the input values of user id and password, then when you click on reset, it will reset input fields with default values.

---

5. <input type="radio">:

The <input> type "radio" defines the radio buttons, which allow choosing an option between a set of related options. At a time only one radio button option can be selected at a time.

Example:

```
1. <form>
```

```
    <p>Kindly Select your favorite color</p>
    <input type="radio" name="color" value="red"> Red <br>
    <input type="radio" name="color" value="blue"> blue <br>
    <input type="radio" name="color" value="green">green <br>
    <input type="radio" name="color" value="pink">pink <br>
    <input type="submit" value="submit">
</form>
```

Output:

Input "radio" type

Kindly Select your favoritecolor

- Red
- blue
- green
- pink

---

6. <input type="checkbox">:

The <input> type "checkbox" are displayed as square boxes which can be checked or unchecked to select the choices from the given options.

Note: The "radio" buttons are similar to checkboxes, but there is an important difference between both types: radio buttons allow the user to select only one option at a time, whereas checkbox allows a user to select zero to multiple options at a time.

Example:

```
<form>
    <label>Enter your Name:</label>
    <input type="text" name="name">
    <p>Kindly Select your favourite sports</p>
    <input type="checkbox" name="sport1" value="cricket">Cricket<br>
    <input type="checkbox" name="sport2" value="tennis">Tennis<br>
    <input type="checkbox" name="sport3" value="football">Football<br>
    <input type="checkbox" name="sport4" value="baseball">Baseball<br>
    <input type="checkbox" name="sport5" value="badminton">Badminton<br><br>

    <input type="submit" value="submit">
</form>
```

Output:

Input "checkbox" type

Registration Form

Enter your Name:

Kindly Select your favorite sports

- Cricket
- Tennis
- Football
- Baseball
- Badminton

---

7. <input type="button">:

The <input> type "button" defines a simple push button, which can be programmed to control a functionally on any event such as, click event.

Note: It mainly works with JavaScript.

Example:

```
<form>
    <input type="button" value="Clcik me " onclick="alert('you are learning HTML')">
</form>
```

Output:

Input "button" type.

Click the button to see the result:

Note: In the above example we have used the "alert" of JS, which you will learn in our JS tutorial. It is used to show a pop window.

---

8. <input type="file">:

The <input> element with type "file" is used to select one or more files from user device storage. Once you select the file, and after submission, this file can be uploaded to the server with the help of JS code and file API.

Example:

```
<form>
    <label>Select file to upload:</label>
```

```
    <input type="file" name="newfile">
    <input type="submit" value="submit">
</form>
```

Output:

Input "file" type.

We can choose any type of file until we do not specify it! The selected file will appear at next to "choose file" option

Select file to upload:

---

9. <input type="image">:

The <input> type "image" is used to represent a submit button in the form of image.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>Input "image" type.</h2>
<p>We can create an image as submit button</p>
  <form>
    <label>User id:</label><br>
     <input type="text" name="name"><br><br>
     <input type="image" alt="Submit" src="login.png"  width="100px">
  </form>

 </body>
</html>
```

HTML5 newly added <input> types element

1. <input type="color">:

The <input> type "color" is used to define an input field which contains a colour. It allows a user to specify the colour by the visual colour interface on a browser.

Note: The "color" type only supports color value in hexadecimal format, and the default value is #000000 (black).

Example:

```
<form>
    Pick your Favorite color: <br><br>
```

```
    <input type="color" name="upclick" value="#a52a2a"> Upclick<br><br>
    <input type="color" name="downclick" value="#f5f5dc"> Downclick
</form>
```

Output:

Input "color" types:
Pick your Favoritecolor:

Up-click

Down-click

Note: The default value of "color" type is #000000 (black). It only supports color value in hexadecimal format.

---

2. <input type="date">:

The <input> element of type "date" generates an input field, which allows a user to input the date in a given format. A user can enter the date by text field or by date picker interface.

Example:

```
<form>
    Select Start and End Date: <br><br>
      <input type="date" name="Startdate"> Start date:<br><br>
      <input type="date" name="Enddate"> End date:<br><br>
      <input type="submit">
</form>
```

Output:

Input "date" type
Select Start and End Date:

Start date:

End date:

---

3. <input type="datetime-local">:

The <input> element of type "datetime-local" creates input filed which allow a user to select the date as well as local time in the hour and minute without time zone information.

Example:

```
<form>
    <label>
```

```
3.        Select the meeting schedule: <br><br>
4.        Select date & time: <input type="datetime-local" name="meetingdate"> <br><br>
5.      </label>
6.        <input type="submit">
7.  </form>
```

Output:

Input "datetime-local" type
Select the meeting schedule:

Select date & time:

---

4. <input type="email">:

The <input> type "email" creates an input filed which allow a user to enter the e-mail address with pattern validation. The multiple attributes allow a user to enter more than one email address.

Example:

```
1.  <form>
2.          <label><b>Enter your Email-address</b></label>
3.          <input type="email" name="email" required>
4.          <input type="submit">
5.           <p><strong>Note:</strong>User can also enter multiple email addresses separat
    ing by comma or whitespace as following: </p>
6.          <label><b>Enter multiple Email-addresses</b></label>
7.           <input type="email" name="email"  multiple>
8.          <input type="submit">
9.  </form>
```

Output:

Input "email" type
Enter your Email-address

Note: User can also enter multiple email addresses separating by comma or whitespace as following:

Enter multiple Email-addresses

---

5. <input type="month">:

The <input> type "month" creates an input field which allows a user to easily enter month and year in the format of "MM, YYYY" where MM defines month value, and YYYY defines the year value. New

Example:

```
<form>
    <label>Enter your Birth Month-year: </label>
    <input type="month" name="newMonth">
    <input type="submit">
</form>
```

Output:

Input "month" type:
Enter your Birth Month-year:

HTML form Attribute

HTML <form> element attributes

In HTML there are various attributes available for <form> element which are given below:

HTML action attribute

The action attribute of <form> element defines the process to be performed on form when form is submitted, or it is a URI to process the form information.

The action attribute value defines the web page where information proceed. It can be .php, .jsp, .asp, etc. or any URL where you want to process your form.

Note: If action attribute value is blank then form will be processed to the same page.
Example:

```
<form action="action.html" method="post">
<label>User Name:</label><br>
<input type="text" name="name"><br><br>
<label>User Password</label><br>
<input type="password" name="pass"><br><br>
 <input type="submit">
   </form>
```

Output:

Demo of action attribute of form element

User Name:

User Password

It will redirect to a new page "action.html" when you click on submit button

---

HTML method attribute

The method attribute defines the HTTP method which browser used to submit the form. The possible values of method attribute can be:

- **post:** We can use the post value of method attribute when we want to process the sensitive data as it does not display the submitted data in URL.

Example:

1. <form action="action.html" method="post">

- **get:** The get value of method attribute is default value while submitting the form. But this is not secure as it displays data in URL after submitting the form.

Example:

1. <form action="action.html" method="get">

When submitting the data, it will display the entered data in the form of:

1. file:///D:/HTML/action.html?name=JavaTPoint&pass=123

HTML target attribute

The target attribute defines where to open the response after submitting the form. The following are the keywords used with the target attribute.

- **_self:** If we use _self as an attribute value, then the response will display in current page only.

Example:

1. <form action="action.html" method="get" target="_self">

- **_blank:** If we use _blank as an attribute it will load the response in a new page.

Example:

```
1. <form action="action.html" method="get" target="_blank">
```

HTML autocomplete attribute

The HTML autocomplete attribute is a newly added attribute of HTML5 which enables an input field to complete automatically. It can have two values "on" and "off" which enables autocomplete either ON or OFF. The default value of autocomplete attribute is "on".

Example:

```
1. <form action="action.html" method="get" autocomplete="on">
```

Example:

```
1. <form action="action.html" method="get" autocomplete="off">
```

Note: it can be used with <form> element and <input> element both.

HTML enctype attribute

The HTML enctype attribute defines the encoding type of form-content while submitting the form to the server. The possible values of enctype can be:

- **application/x-www-form-urlencoded:** It is default encoding type if the enctype attribute is not included in the form. All characters are encoded before submitting the form.

Example:

```
1. <form action="action.html" method="post" enctype="application/x-www-form-urlencoded" >
```

- **multipart/form-data:** It does not encode any character. It is used when our form contains file-upload controls.

Example:

```
1. <form action="action.html" method="post" enctype="multipart/form-data">
```

- **text/plain (HTML5):** In this encoding type only space are encoded into + symbol and no any other special character encoded.

Example:

```
1. <form action="action.html" method="post" enctype="text/plain" >
```

HTML novalidate attribute HTML5

The novalidate attribute is newly added Boolean attribute of HTML5. If we apply this attribute in form then it does not perform any type of validation and submit the form.

Example:

```
1. <form action = "action.html" method = "get" novalidate>
```

Output:

Fill the form

Enter name:

Enter age:

Enter email:

Try to change the form detials with novalidateatttribute and without novalidate attribute and see the difference.

HTML <input> element attribute

HTML name attribute

The HTML name attribute defines the name of an input element. The name and value attribute are included in HTTP request when we submit the form.

Note: One should not omit the name attribute as when we submit the form the HTTP request includes both name-value pair and if name is not available it will not process that input field.

Example:

```
1. <form action = "action.html" method = "get">
2.         Enter name:<br><input type="name" name="uname"><br>
3.          Enter age:<br><input type="number" name="age"><br>
4.         Enter email:<br><input type="email"><br>
5.          <input type="submit" value="Submit">
6.       </form>
```

Output:

Fill the form

Enter name:

Enter age:

Enter email:

Note: If you will not use name attribute in any input field, then that input field will not be submitted, when submit the form.

Click on submit and see the URL where email is not included in HTTP request as we have not used name attribute in the email input field

---

HTML value attribute

The HTML value attribute defines the initial value or default value of an input field.

Example:

```
<form>
        <label>Enter your Name</label><br>
         <input type="text" name="uname" value="Enter Name"><br><br>
        <label>Enter your Email-address</label><br>
         <input type="text" name="uname" value="Enter email"><br><br>
          <label>Enter your password</label><br>
         <input type="password" name="pass" value=""><br><br>
        <input type="submit" value="login">
    </form>
```

Output:

Fill the form

Enter your Name
Enter Name

Enter your Email-address
Enter email

Enter your password

Note: In password input filed the value attribute will always unclear

HTML required attribute HTML5

HTML required is a Boolean attribute which specifies that user must fill that filed before submitting the form.

Example:

```
<form>
      <label>Enter your Email-address</label><br>
       <input type="text" name="uname" required><br><br>
       <label>Enter your password</label><br>
       <input type="password" name="pass"><br><br>
      <input type="submit" value="login">
 </form>
```

Output:

Fill the form

Enter your Email-address

Enter your password

If you will try to submit the form without completing email field then it will give an error pop up.

HTML autofocus attribute HTML5

The autofocus is a Boolean attribute which enables a field automatically focused when a webpage loads.

Example:

```
<form>
      <label>Enter your Email-address</label><br>
       <input type="text" name="uname" autofocus><br><br>
       <label>Enter your password</label><br>
       <input type="password" name="pass"><br><br>
      <input type="submit" value="login">
 </form>
```

HTML placeholder attribute HTML5

The placeholder attribute specifies a text within an input field which informs the user about the expected input of that filed.

The placeholder attribute can be used with text, password, email, and URL values.

When the user enters the value, the placeholder will be automatically removed.

Example:

```
<form>
      <label>Enter your name</label><br>
       <input type="text" name="uname" placeholder="Your name"><br><br>
         <label>Enter your Email address</label><br>
       <input type="email" name="email" placeholder="example@gmail.com"><br><
br>
         <label>Enter your password</label><br>
       <input type="password" name="pass" placeholder="your password"><br><br>
      <input type="submit" value="login">
    </form>
```

Output:

Registration form
Enter your name

Enter your Email address

Enter your password

HTML disabled attribute

The HTML disabled attribute when applied then it disable that input field. The disabled field does not allow the user to interact with that field.

The disabled input filed does not receive click events, and these input value will not be sent to the server when submitting the form.

Example:

```
<input type="text" name="uname" disabled><br><br>
```

Output:

Registration form
Enter User name
USER

Enter your Email address

Enter your password

HTML size attribute

The size attribute controls the size of the input field in typed characters.

Example:

```
<label>Account holder name</label><br>
        <input type="text" name="uname" size="40" required><br><br>
         <label>Account number</label><br>
        <input type="text" name="an" size="30" required><br><br>
         <label>CVV</label><br>
        <input type="text" name="cvv"  size="1" required><br><br>
```

Output:

Registration form with disbaled attribute
Account holder name

Account number

CVV

HTML form attribute

HTML form attribute allows a user to specify an input filed outside the form but remains the part of the parent form.

Example:

```
1. User email: <br><input type="email" name="email"  form="fcontrol"  required><br>
2.          <input type="submit" form="fcontrol">
```

Output:

User Name:

User password:

The email field is outside the form but still it will remain part of the form

User email:

HTML style using CSS

Let's suppose we have created our web page using a simple HTML code, and we want something which can present our page in a correct format, and visibly attractive. So to do this, we can style our web page with CSS (Cascading Stylesheet) properties.

CSS is used to apply the style in the web page which is made up of HTML elements. It describes the look of the webpage.

CSS provides various style properties such as background color, padding, margin, border-color, and many more, to style a webpage.

Each property in CSS has a name-value pair, and each property is separated by a semicolon (;).

Note: In this chapter, we have given a small overview of CSS. You will learn everything in depth about CSS in our CSS tutorial.

Example:

```
1. <body style="text-align: center;">
2.       <h2 style="color: red;">Welcome to javaTpoint</h2>
3.       <p style="color: blue; font-size: 25px; font-
   style: italic ;">This is a great website to learn technologies in very simple way. </p>
4. </body>
```

In the above example, we have used a style attribute to provide some styling format to our code.

Output:

Welcome to javaTpoint

This is a great website to learn technologies in very simple way.

Three ways to apply CSS

To use CSS with HTML document, there are three ways:

- **Inline CSS:** Define CSS properties using style attribute in the HTML elements.
- **Internal or Embedded CSS:** Define CSS using <style> tag in <head> section.
- **External CSS:** Define all CSS property in a separate .css file, and then include the file with HTML file using tag in section.

Inline CSS:

Inline CSS is used to apply CSS in a single element. It can apply style uniquely in each element.

To apply inline CSS, you need to use style attribute within HTML element. We can use as many properties as we want, but each property should be separated by a semicolon (;).

Example:

```
<h3 style="color: red;
        font-style: italic;
         text-align: center;
        font-size: 50px;
         padding-top: 25px;">Learning HTML using Inline CSS</h3>
```

Output:

Learning HTML using Inline CSS

Internal CSS:

An Internal stylesheets contains the CSS properties for a webpage in <head> section of HTML document. To use Internal CSS, we can use class and id attributes.

We can use internal CSS to apply a style for a single HTML page.

Example:

```
<!DOCTYPE html>
```

```
<html>
<head>
                <style>
      /*Internal CSS using element name*/
          body{background-color:lavender;
            text-align: center;}
           h2{font-style: italic;
             font-size: 30px;
           color: #f08080;}
           p{font-size: 20px;}
       /*Internal CSS using class name*/
           .blue{color: blue;}
          .red{color: red;}
           .green{color: green;}
     </style>
   </head>
  <body>
   <h2>Learning HTML with internal CSS</h2>
   <p class="blue">This is a blue color paragraph</p>
   <p class="red">This is a red color paragraph</p>
   <p class="green">This is a green color paragraph</p>
  </body>
</html>
```

Note: In the above example, we have used a class attribute which you will learn in the next chapter.

External CSS:

An external CSS contains a separate CSS file which only contains style code using the class name, id name, tag name, etc. We can use this CSS file in any HTML file by including it in HTML file using <link> tag.

If we have multiple HTML pages for an application and which use similar CSS, then we can use external CSS.

There are two files need to create to apply external CSS

- First, create the HTML file
- Create a CSS file and save it using the .css extension (This file only will only contain the styling code.)
- Link the CSS file in your HTML file using tag in header section of HTML document.

Example:

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

```
7.      <h2>Learning HTML with External CSS</h2>
8.       <p class="blue">This is a blue color paragraph</p>
9.       <p class="red">This is a red color paragraph</p>
10.     <p class="green">This is a green color paragraph</p>
11.   </body>
12.  </html>
```

CSS file:

```
body{
background-color:lavender;
text-align: center;
}
h2{
font-style: italic;
size: 30px;
color: #f08080;
}
p{
font-size: 20px;
}

.blue{
color: blue;
}
.red{
color: red;
}
.green{
color: green;
}
```

Commonly used CSS properties:

Properties-name	Syntax	Description
background-color	background-color:red;	It defines the background color of that element.
color	color: lightgreen;	It defines the color of text of an element
padding	padding: 20px;	It defines the space between content and the border.
margin	margin: 30px; margin-left:	It creates space around an element.
font-family	font-family: cursive;	Font-family defines a font for a particular element.
Font-size	font-size: 50px;	Font-size defines a font size for a particular element.
text-align	text-align: left;	It is used to align the text in a selected position.

HTML Classes

Class Attribute in HTML

The HTML class attribute is used to specify a single or multiple class names for an HTML element. The class name can be used by CSS and JavaScript to do some tasks for HTML elements. You can use this class in CSS with a specific class, write a period (.) character, followed by the name of the class for selecting elements.

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

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

Defining an HTML class

To create an HTML class, firstly define style for HTML class using <style> tag within <head> section as following example:

Example:

```
<head>
   <style>
      .headings{
         color: lightgreen;
          font-family: cursive;
         background-color: black; }
   </style>
</head>
```

We have define style for a class name "headings", and we can use this class name with any of HTML element in which we want to provide such styling. We just need to follow the following syntax to use it.

```
<tag class="ghf"> content </tag>
```

Example 1:

```
<!DOCTYPE html>
<html>
<head>
   <style>
      .headings{
         color: lightgreen;
          font-family: cursive;
         background-color: black; }
   </style>
</head>
<body>
<h1 class="headings">This is first heading</h1>
```

```
<h2 class="headings">This is Second heading</h2>
<h3 class="headings">This is third heading</h3>
<h4 class="headings">This is fourth heading</h4>
</body>
</html>
```

Another Example with different class name

Example:

Let's use a class name "Fruit" with CSS to style all elements.

```
<style>
.fruit {
    background-color: orange;
    color: white;
    padding: 10px;
}
</style>

<h2 class="fruit">Mango</h2>
<p>Mango is king of all fruits.</p>

<h2 class="fruit">Orange</h2>
<p>Oranges are full of Vitamin C.</p>

<h2 class="fruit">Apple</h2>
<p>An apple a day, keeps the Doctor away.</p>
```

Here you can see that we have used the class name "fruit" with (.) to use all its elements.

Note: You can use class attribute on any HTML element. The class name is case-sensitive.

Class Attribute in JavaScript

You can use JavaScript access elements with a specified class name by using the getElementsByClassName() method.

Example:

Let's hide all the elements with class name "fruit" when the user click on the button.

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

<h2>Class Attribute with JavaScript</h2>
```

```
<p>Click the button, to hide all elements with the class name "fruit", with JavaScript:</p>

<button onclick="myFunction()">Hide elements</button>


<h2 class="fruit">Mango</h2>
<p>Mango is king of all fruits.</p>

<h2 class="fruit">Orange</h2>
<p>Oranges are full of Vitamin C.</p>

<h2 class="fruit">Apple</h2>
<p>An apple a day, keeps the Doctor away.</p>

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

</body>
</html>
```

Note: You will learn more about JavaScript in our JavaScript tutorial.

Multiple Classes

You can use multiple class names (more than one) with HTML elements. These class names must be separated by a space.

Example:

Let's style elements with class name "fruit" and also with a class name "center".

```
<!DOCTYPE html>
<html>
<style>
.fruit {
    background-color: orange;
    color: white;
    padding: 10px;
}

.center {
    text-align: center;
}
</style>
```

```
<body>

<h2>Multiple Classes</h2>
<p>All three elements have the class name "fruit". In addition, Mango also have the cl
ass name "center", which center-aligns the text.</p>

<h2 class="fruit center">Mango</h2>
<h2 class="fruit">Orange</h2>
<h2 class="fruit">Apple</h2>

</body>
</html>
```

You can see that the first element <h2> belongs to both the "fruit" class and the "center" class.

Same class with Different Tag

You can use the same class name with different tags like <h2> and <p> etc. to share the same style.

Example:

```
<!DOCTYPE html>
<html>
<style>
.fruit {
  background-color: orange;
  color: white;
  padding: 10px;
}
</style>
<body>
<h2>Same Class with Different Tag</h2>
<h2 class="fruit">Mango</h2>
<p class="fruit">Mango is the king of all fruits.</p>
</body>
</html>
```

Test it Now**HTML Id Attribute**

The **id attribute** is used to specify the unique ID for an element of the HTML document. It allocates the unique identifier which is used by the **CSS** and the **JavaScript** for performing certain tasks.

Note: In the Cascading Style sheet (CSS), we can easily select an element with the specific id by using the # symbol followed by id.

Note: JavaScript can access an element with the given ID by using the getElementById() method.

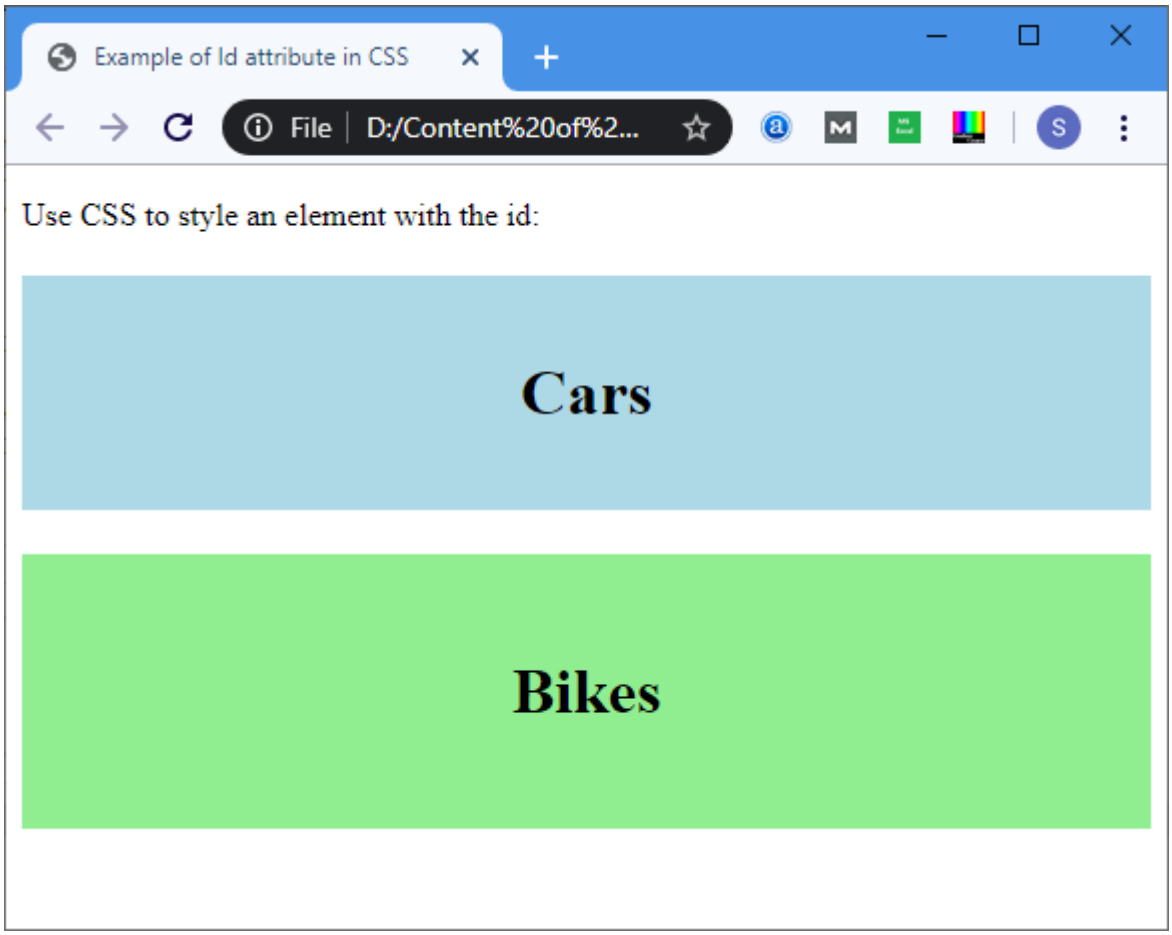
Syntax

```
1. <tag id="value">
```

Example 1: The following example describes how to use the id attribute in CSS document:

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>
5. Example of Id attribute in CSS
6. </title>
7. <style>
8. #Cars {
9. padding: 40px;
10. background-color: lightblue;
11. color: black;
12. text-align: center;
13. }
14.
15. #Bikes
16. {
17. padding: 50px;
18. background-color: lightGreen;
19. text-align: center;
20. }
21. </style>
22. </head>
23. <body>
24. <p> Use CSS to style an element with the id: </p>
25. <h1 id="Cars"> Cars </h1>
26. <h1 id="Bikes"> Bikes </h1>
27. </body>
28. </html>
```

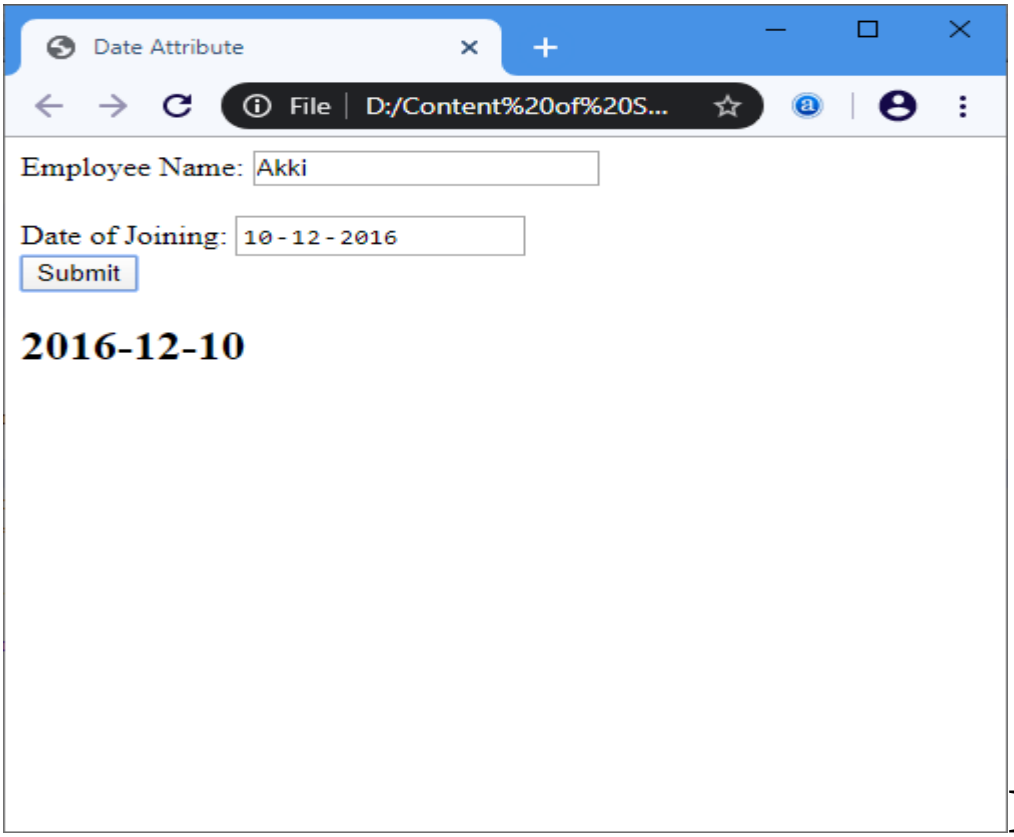
Output:



Example 2: The following example describes how to use the ID attribute in JavaScript.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title> Date Attribute </title>
5. <script>
6. function viewdate() {
7. var x = document.getElementById("dob").value;
8. document.getElementById("demo").innerHTML = x;
9. </script>
10. </head>
11. <body>
12. Employee Name: <input type="text" placeholder="Your Good name"/>
13. <br>
14. <br>
15. Date of Joining:
16. <input type="date" id="dob">
17. <br>
18. <button onclick="viewdate()"> Submit
19. </button>
20. <br>
21. <h2 id="demo"> </h2>
22. </body>
23. </html>
```

Output:



HTML List Box

The **list box** is a graphical control element in the HTML document that allows a user to select one or more options from the list of options.

Syntax

To create a list box, use the HTML element**<select>** which contains two attributes **Name** and **Size**. The **Name** attribute is used to define the name for calling the list box, and **size** attribute is used to specify the numerical value that shows the how many options it contains.

```
<select Name="Name_of_list_box" Size="Number_of_options">
  <option> List item 1 </option>
  <option> List item 2 </option>
  <option> List item 3 </option>
  <option> List item N </option>
</select>
```

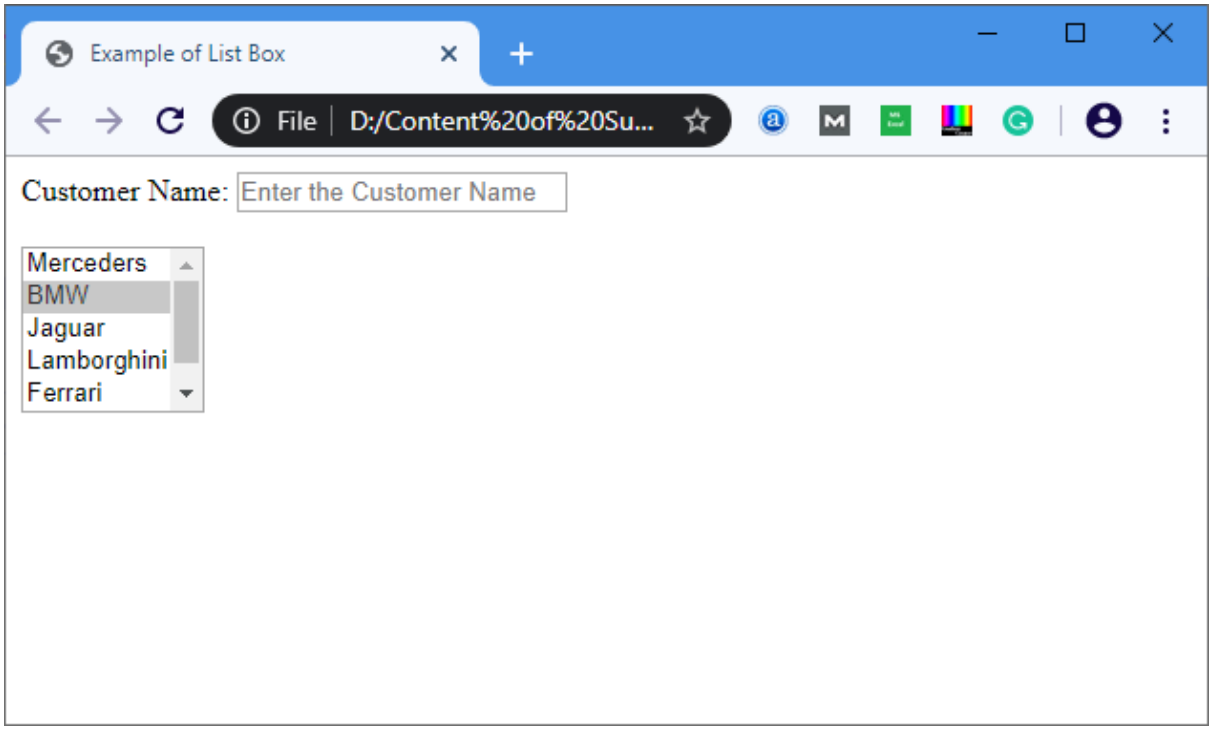
Examples:

Example 1: Consider the below example that creates a simple list box.

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

```
<title>
Example of List Box
</title>
<body>
Customer Name:  <input type="text" Placeholder="Enter the Customer Name"/>
<br>
<br>
<select name="Cars" size="5">
    <option value="Merceders"> Merceders </option>
    <option value="BMW"> BMW </option>
    <option value="Jaguar"> Jaguar </option>
    <option value="Lamborghini"> Lamborghini </option>
    <option value="Ferrari"> Ferrari </option>
    <option value="Ford"> Ford </option>
</select>
</body>
</html>
```

Output:

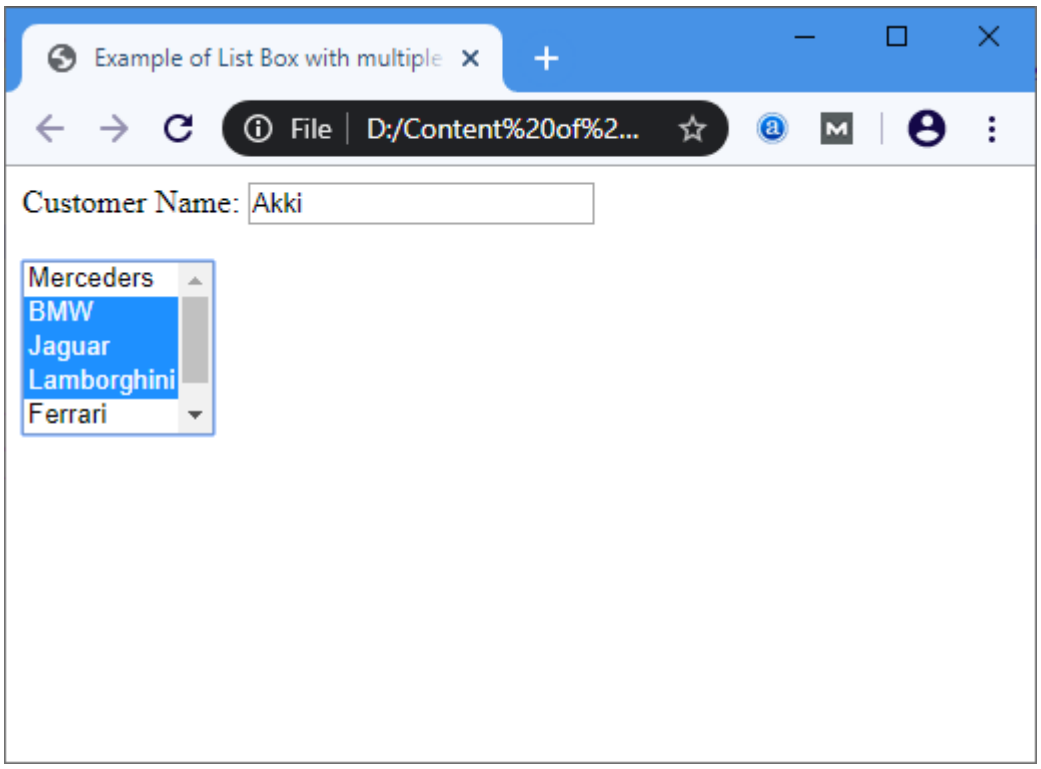


Example 2: Below example uses the **multiple**attribute for selecting the multiple options in a list. We can select multiple options from list box by holding the ctrl key.

```
<!DOCTYPE html>
<html>
<title>
Example of List Box with multiple attribute
</title>
<body>
Customer Name:  <input type="text" Placeholder="Enter the Customer Name"/>
```

```
8.  <br>
9.  <br>
10. <select name="Cars" size="5" multiple="multiple">
11.     <option value="Merceders"> Merceders </option>
12.     <option value="BMW"> BMW </option>
13.     <option value="Jaguar"> Jaguar </option>
14.     <option value="Lamborghini"> Lamborghini </option>
15.     <option value="Ferrari"> Ferrari </option>
16.     <option value="Ford"> Ford </option>
17. </select>
18. </body>
19. </html>
```

Output:



Unix commands list

This guide has been prepared by me to help myself with the list of frequently used ***basic commands in UNIX/LINUX*** to be on my finger tip. Thought of sharing it with the others, in case, it might turn out helpful to other readers as well. This is ***Unix/Linux basic commands - 1***,for 2nd part follow the link given at the end of this article.

Web servers Shell

Unix/Linux file commands guide

This article will serve as a 5 minute guide or tutorial to learn/revisit basic unix or linux commands frequently used while working with files. Unix/Linux command is given along with their *usage or description*.

- **ls ►** use this *command in unix/linux* to see all the directory listing. However, any hidden files will not be listed.
- **ls -al ►** use this *command in unix/linux* to see formatted directory listing along with the hidden files.
- **ls -lt ►** use this *command in unix/linux* to sort the directory listing by their time of modification.
- **pwd ►** use this *command in unix/linux* to show your current working directory.
- **touch fileName ►** use this *command in unix/linux* to create new file with its name as filename.
- **cd ►** use this *command in unix/linux* to move to home directory.
- **cd dirName ►** use this *command in unix/linux* to change current directory to dirName directory.
- **mkdirdirName ►** use this *command in unix/linux* to make or create directory having name as dirName.
- **rm fileName ►** use this *command in unix/linux* to remove or delete file having name as fileName.
- **rm -r dirName ►** use this *command in unix/linux* to remove or delete directory dirName.
- **rm -f filename ►** use this *command in unix/linux* to force remove the file filename.
- **more fileName ►** use this *command in unix/linux* to get the content of file having name as filename
- **head fileName ►** use this *command in unix/linux* to get output of first 10 lines of the file fileName.
- **tail fileName ►** use this *command in unix/linux* to get output of last 10 lines of the file filename.
- **cp fileAfileB ►** use this *command in unix/linux* to copy the content of fileA to fileB.
- **cp -r dirAdirB ►** use this *command in unix/linux* to copy directory dirA to directory dirB and create dirB if not already created.
- **mv fileAfileB ►** use this *command in unix/linux* to rename or move fileA to fileB.
- **cat >file ►** use this *command in unix/linux* to place standard input into the file.

Unix or Linux process management commands guide

This section will serve as a 5 minute guide or tutorial to learn/revisit basic unix or linux commands frequently used while working with process management. Unix/Linux command is given along with their *usage or description*.

- **ps ►** use this command in unix/linux to see currently working processes.
- **top ►** use this command in unix/linux to display all the running processes.

- **kill pid ►** use this command in unix/linux to kill the process with given pid.
- **killallprocessA ►** use this command in unix/linux to kill all the process named as processA
- **pkill pattern ►** use this command in unix/linux to kill all processes matching the given pattern.
- **bg ►** use this command in unix/linux to list all the background jobs.
- **fg ►** use this command in unix/linux to bring the most recent job to foreground.
- **fg n1 ►** use this command in unix/linux to bring job n1 to the foreground.

Unix/Linux system info commands guide

This section will serve as a 5 minute guide or tutorial to learn/revisit basic unix or linux commands frequently used while working with system. Unix/Linux command is given along with their *usage or description*.

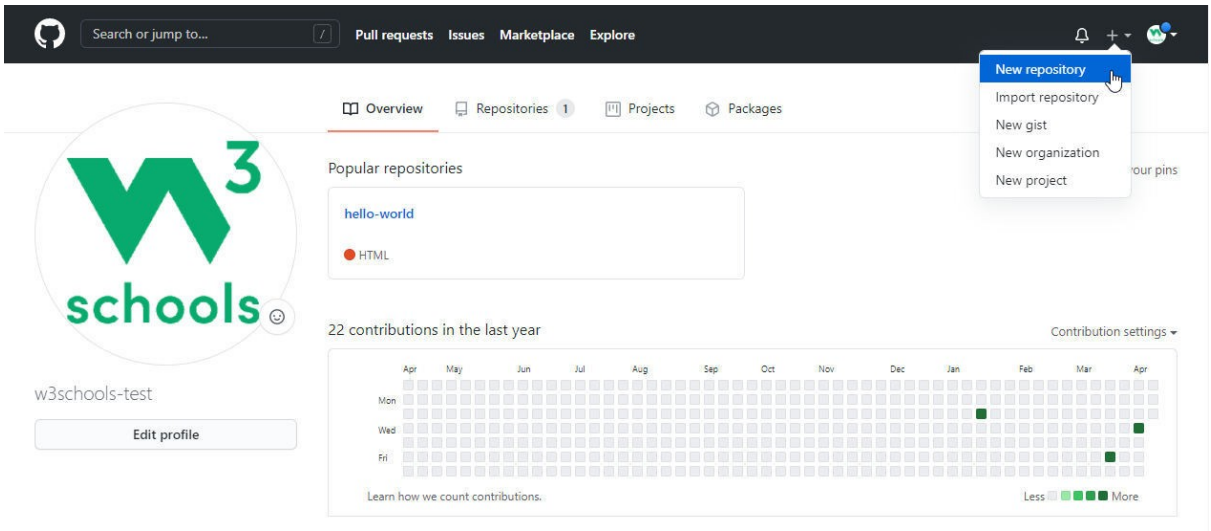
- **cal ►** use this command in unix/linux to show current months calendar.
- **date ►** use this command in unix/linux to show current date and time.
- **w ►** use this command in unix/linux to see who all are currently logged in to the system.
- **whoami ►** use this command in unix/linux to see who you are currently logged in as in the system.
- **uname -a ►** use this command in unix/linux to see kernel information.
- **finger user ►** use this command in unix/linux to display information about user.
- **man command ►** use this command in unix/linux to show the manual for command.
- **free ►** use this command in unix/linux to show memory and swap usage.
- **df ►** use this command in unix/linux to see the disk usage.
- **du ►** use this command in unix/linux to see the directory space usage.
- **whereis app ►** use this command in unix/linux to show possible location of app.
- **which app ►** use this command in unix/linux to show which application will be run by default.

Version control

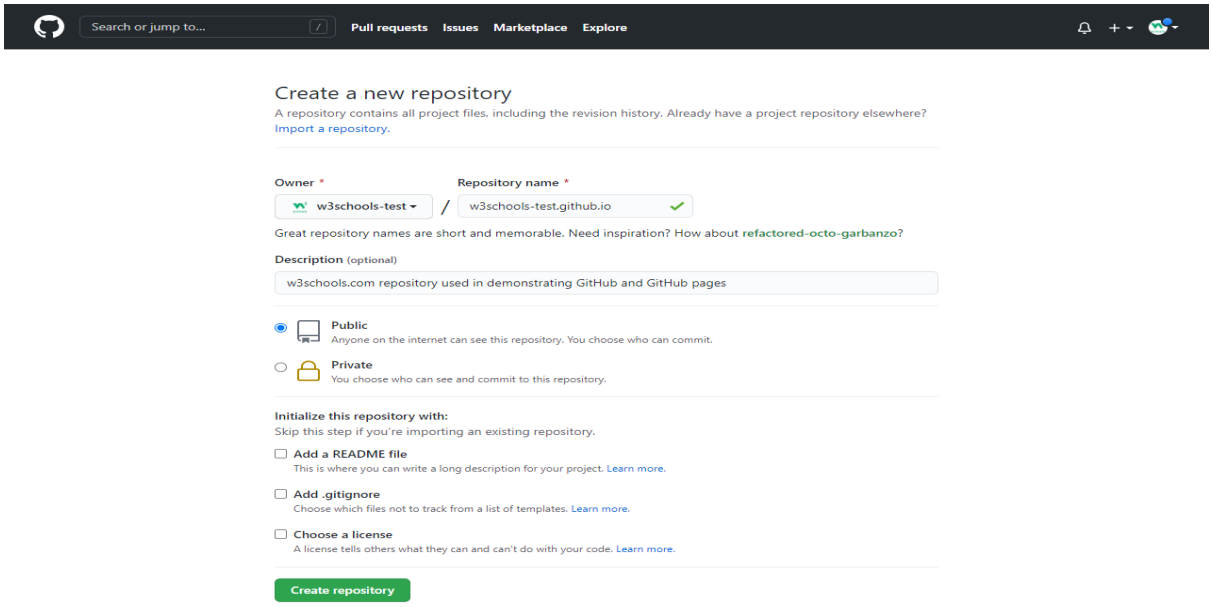
Git GitHub Pages

Create a New Repository

Start by signing in to GitHub. GitHub pages need a special name and setup to work, so we start by creating a new repository:



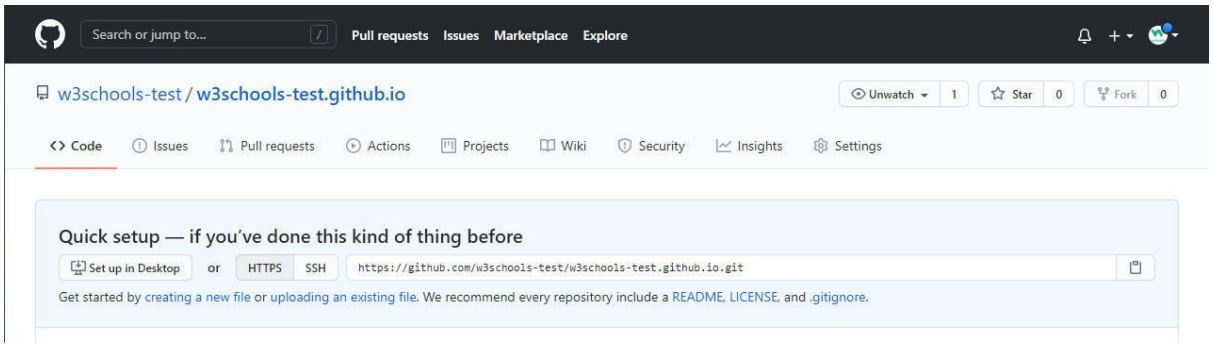
This repository needs a special name to function as a GitHub page. It needs to be your GitHub username, followed by .github.io:



Push Local Repository to GitHub Pages

We add this new repository as a remote for our local repository, we are calling it gh-page (for GitHub Pages).

Copy the URL from here:



And add it as a new remote:

Example

```
git remote addgh-page https://github.com/w3schools-test/w3schools-test.github.io.git
```

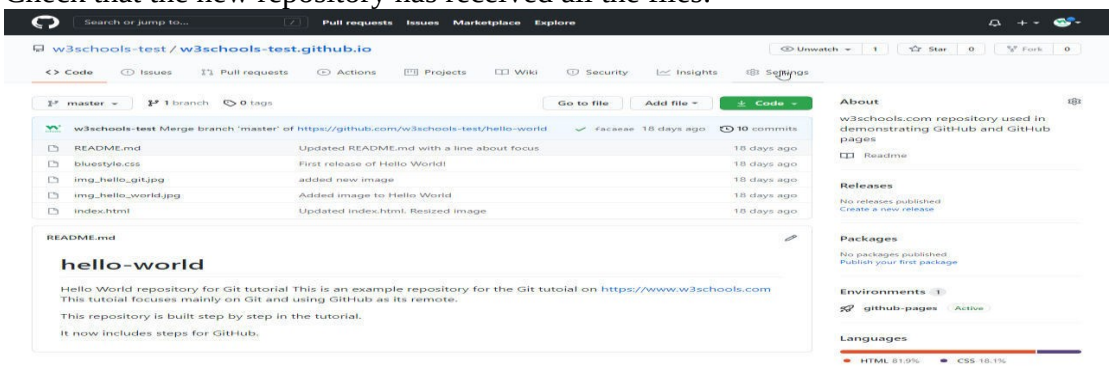
Make sure you are on the masterbranch, then push the masterbranch to the new remote:

Example

```
git push gh-page master
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 16 threads
Compressing objects: 100% (33/33), done.
Writing objects: 100% (33/33), 94.79 KiB | 15.80 MiB/s, done.
Total 33 (delta 18), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (18/18), done.
To https://github.com/w3schools-test/w3schools-test.github.io.git
 * [new branch]      master -> master
```

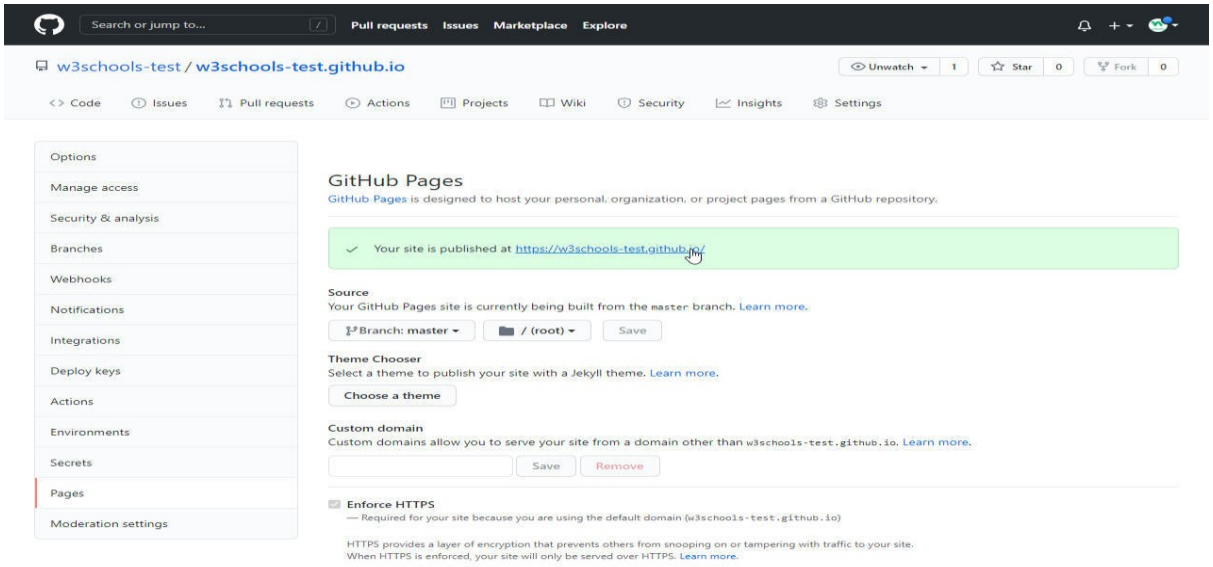
Note: If this is the first time you are connecting to GitHub, you will get some kind of notification to authenticate this connection.

Check that the new repository has received all the files:



Check Out Your Own GitHub Page

That looks good, now click the Settings menu and navigate to the Pages tab:



The GitHub page is created, and you can click the URL to view the result!

Git Tutorial

Learning by Examples

In this tutorial, we will show you Git commands like this:

Example

```
git --version
git version 2.30.2.windows.1
```

For new users, using the terminal view can seem a bit complicated. Don't worry! We will keep it really simple, and learning this way gives you a good grasp of how Git works.

In the code above, you can see commands (input) and output.

Lines like this are commands we input:

Example

```
git --version
```

Lines like this are the output/response to our commands:

Example

```
git version 2.30.2.windows.1
```

In general, lines with $ in front of it is input. These are the commands you can copy and run in your terminal.

Git and Remote Repositories

Git and GitHub are different things.

In this tutorial you will understand what Git is and how to use it on the remote repository platforms, like GitHub.

You can choose, and change, which platform to focus on by clicking in the menu on the right:

Git Exercises

Test Yourself With Exercises

Exercise:

Insert the missing part of the command to check which version of Git (if any) is installed.

git

What is Git?

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

What does Git do?

- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project

Working with Git

- Initialize Git on a folder, making it a **Repository**
- Git now creates a hidden folder to keep track of changes in that folder

- When a file is changed, added or deleted, it is considered **modified**
- You select the modified files you want to **Stage**
- The **Staged** files are **Committed**, which prompts Git to store a **permanent** snapshot of the files
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.
- Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

Why Git?

- Over 70% of developers use Git!
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.

What is GitHub?

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.
- In this tutorial, we will focus on using Git with GitHub.

Git Install

You can download Git for free from the following website: https://www.git-scm.com/

Using Git with Command Line

To start using Git, we are first going to open up our Command shell.

For Windows, you can use Git bash, which comes included in Git for Windows. For Mac and Linux you can use the built-in terminal.

The first thing we need to do, is to check if Git is properly installed:

Example

```
git --version
git version 2.30.2.windows.1
```

If Git is installed, it should show something like `git version X.Y`

Configure Git

Now let Git know who you are. This is important for version control systems, as each Git commit uses this information:

Example
```
git config --global user.name "w3schools-test"
git config --global user.email"test@w3schools.com"
```

Change the user name and e-mail address to your own. You will probably also want to use this when registering to GitHub later on.

Note: Use `global` to set the username and e-mail for **every repository** on your computer.

If you want to set the username/e-mail for just the current repo, you can remove `global`

Creating Git Folder

Now, let's create a new folder for our project:

Example
```
mkdirmyproject
cdmyproject
```

mkdir**make**s a **new directory**.

cd**changes** the **current working directory**.

Now that we are in the correct directory. We can start by initializing Git!

Note: If you already have a folder/directory you would like to use for Git:

Navigate to it in command line, or open it in your file explorer, right-click and select "Git Bash here"

Initialize Git

Once you have navigated to the correct folder, you can initialize Git on that folder:

Example
```
gitinit
Initialized empty Git repository in /Users/user/myproject/.git/
```

You just created your first Git Repository!

Note: Git now knows that it should watch the folder you initiated it on.

Git creates a hidden folder to keep track of changes.

Git Adding New Files

You just created your first local Git repo. But it is empty.

So let's add some files, or create a new file using your favourite text editor. Then save or move it to the folder you just created.

If you want to learn how to create a new file using a text editor, you can visit our HTML tutorial:
HTML Editors

For this example, I am going to use a simple HTML file like this:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
</head>
<body>

<h1>Hello world!</h1>
<p>This is the first file in my new Git Repo.</p>

</body>
</html>
```

And save it to our new folder as index.html.

Let's go back to the terminal and list the files in our current working directory:

Example

```
ls
index.html
```

ls will **list** the files in the directory. We can see that index.html is there.

Then we check the Git status and see if it is a part of our repo:

Example

```
git status
On branch master

No commits yet

Untracked files:
  (use "git add ..." to include in what will be committed)
    index.html

nothing added to commit but untracked files present (use "git add" to track)
```

Now Git is **aware** of the file, but has not **added** it to our repository!

Files in your Git repository folder can be in one of 2 states:

- Tracked - files that Git knows about and are added to the repository
- Untracked - files that are in your working directory, but not added to the repository

When you first add files to an empty repository, they are all untracked. To get Git to track them, you need to stage them, or add them to the staging environment.

Git Staging Environment

One of the core functions of Git is the concepts of the Staging Environment, and the Commit.

As you are working, you may be adding, editing and removing files. But whenever you hit a milestone or finish a part of the work, you should add the files to a Staging Environment.

Staged files are files that are ready to be **committed** to the repository you are working on. You will learn more about commit shortly.

For now, we are done working with index.html. So we can add it to the Staging Environment:

Example
```
gitadd index.html
```

The file should be **Staged**. Let's check the status::

Example
```
git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached ..." to unstage)
    new file: index.html
```

Now the file has been added to the Staging Environment.

Git Add More than One File

You can also stage more than one file at a time. Let's add 2 more files to our working folder. Use the text editor again.

A README.md file that describes the repository (recommended for all repositories):

Example

```
# hello-world
Hello World repository for Git tutorial
This is an example repository for the Git tutoial on https://www.w3schools.com

This repository is built step by step in the tutorial.
```

A basic external style sheet (bluestyle.css):

Example

```
body {
background-color: lightblue;
}

h1 {
color: navy;
margin-left: 20px;
}
```

And update index.html to include the stylesheet:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
<linkrel="stylesheet"href="bluestyle.css">
</head>
<body>

<h1>Hello world!</h1>
<p>This is the first file in my new Git Repo.</p>

</body>
</html>
```

Now add all files in the current directory to the Staging Environment:

Example

```
gitadd --all
```

Using --all instead of individual filenames will stage all changes (new, modified, and deleted) files.

Example

```
git status
On branch master

No commits yet

Changes to be committed:
```

```
(use "git rm --cached ..." to unstage)
      new file:   README.md
      new file:   bluestyle.css
      new file:   index.html
```

Now all 3 files are added to the Staging Environment, and we are ready to do our first commit.

Git Commit

Since we have finished our work, we are ready move from stage to commit for our repo.

Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.

When we commit, we should **always** include a **message**.

By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when.

Example

```
git commit -m "First release of Hello World!"
[master (root-commit) 221ec6e] First release of Hello World!
 3 files changed, 26 insertions(+)
 create mode 100644 README.md
 create mode 100644 bluestyle.css
 create mode 100644 index.html
```

The commit command performs a commit, and the -m "*message*" adds a message.

The Staging Environment has been committed to our repo, with the message:
"First release of Hello World!"

Git Commit without Stage

Sometimes, when you make small changes, using the staging environment seems like a waste of time. It is possible to commit changes directly, skipping the staging environment. The -a option will automatically stage every changed, already tracked file.

Let's add a small update to index.html:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
<linkrel="stylesheet"href="bluestyle.css">
</head>
<body>
```

```
<h1>Hello world!</h1>
<p>This is the first file in my new Git Repo.</p>
<p>A new line in our file!</p>

</body>
</html>
```

And check the status of our repository. But this time, we will use the --short option to see the changes in a more compact way:

Example

```
git status --short
 M index.html
```

Git Help

If you are having trouble remembering commands or options for commands, you can use Git help.

There are a couple of different ways you can use the help command in command line:

- git *command* -help- See all the available options for the specific command
- git help --all- See all possible commands

Let's go over the different commands.

Git -help See Options for a Specific Command

Any time you need some help remembering the specific option for a command, you can use git *command* -help:

Example

```
git commit -help
usage: git commit [] [--] ...

    -q, --quiet           suppress summary after successful commit
    -v, --verbose         show diff in commit message template

Commit message options
    -F, --file      read message from file
    --author        override author for commit
    --date          override date for commit
    -m, --message
                        commit message
    -c, --reedit-message
                        reuse and edit message from specified commit
    -C, --reuse-message
                        reuse message from specified commit
    --fixup         use autosquash formatted message to fixup specified commit
```

```
    --squash      use autosquash formatted message to squash specified commit
    --reset-author        the commit is authored by me now (used with -C/-c/--amend)
    -s, --signoff         add a Signed-off-by trailer
    -t, --template
                          use specified template file
    -e, --edit            force edit of commit
    --cleanup       how to strip spaces and #comments from message
    --status              include status in commit message template
    -S, --gpg-sign[=]
                          GPG sign commit

Commit contents options
    -a, --all             commit all changed files
    -i, --include         add specified files to index for commit
    --interactive         interactively add files
    -p, --patch           interactively add changes
    -o, --only            commit only specified files
    -n, --no-verify       bypass pre-commit and commit-msg hooks
    --dry-run             show what would be committed
    --short               show status concisely
    --branch              show branch information
    --ahead-behind        compute full ahead/behind values
    --porcelain           machine-readable output
    --long                show status in long format (default)
    -z, --null            terminate entries with NUL
    --amend               amend previous commit
    --no-post-rewrite     bypass post-rewrite hook
    -u, --untracked-files[=]
                          show untracked files, optional modes: all, normal, no. (Default: all)
    --pathspec-from-file
                          read pathspec from file
    --pathspec-file-nul   with --pathspec-from-file, pathspec elements are separated with NUL character
```

Note: You can also use `--help` instead of `-help` to open the relevant Git manual page

---

Git help --all See All Possible Commands

To list all possible commands, use the `help --all` command:

Warning: This will display a very long list of commands

Example

```
$ githelp --all
See 'git help ' to read about a specific subcommand

Main Porcelain Commands
   add              Add file contents to the index
   am               Apply a series of patches from a mailbox
   archive           Create an archive of files from a named tree
   bisect           Use binary search to find the commit that introduced a bug
   branch            List, create, or delete branches
   bundle            Move objects and refs by archive
   checkout          Switch branches or restore working tree files
```

```
   cherry-pick         Apply the changes introduced by some existing commits
citool              Graphical alternative to git-commit
   clean               Remove untracked files from the working tree
   clone               Clone a repository into a new directory
   commit              Record changes to the repository
   describe            Give an object a human readable name based on an available ref
   diff                Show changes between commits, commit and working tree, etc
   fetch               Download objects and refs from another repository
   format-patch        Prepare patches for e-mail submission
gcCleanup unnecessary files and optimize the local repository
gitk                The Git repository browser
   grep                Print lines matching a pattern
gui                 A portable graphical interface to Git
init                Create an empty Git repository or reinitialize an existing one
   log                 Show commit logs
   maintenance         Run tasks to optimize Git repository data
   merge               Join two or more development histories together
   mv                  Move or rename a file, a directory, or a symlink
   notes               Add or inspect object notes
   pull                Fetch from and integrate with another repository or a local branch
   push                Update remote refs along with associated objects
   range-diff          Compare two commit ranges (e.g. two versions of a branch)
   rebase              Reapply commits on top of another base tip
   reset               Reset current HEAD to the specified state
   restore             Restore working tree files
   revert              Revert some existing commits
   rm                  Remove files from the working tree and from the index
shortlog            Summarize 'git log' output
   show                Show various types of objects
   sparse-checkout     Initialize and modify the sparse-checkout
   stash               Stash the changes in a dirty working directory away
   status              Show the working tree status
   submodule           Initialize, update or inspect submodules
   switch              Switch branches
   tag                 Create, list, delete or verify a tag object signed with GPG
worktree            Manage multiple working trees

Ancillary Commands / Manipulators
   config              Get and set repository or global options
   fast-export         Git data exporter
   fast-import         Backend for fast Git data importers
   filter-branch       Rewrite branches
mergetool           Run merge conflict resolution tools to resolve merge conflicts
   pack-refs           Pack heads and tags for efficient repository access
   prune               Prune all unreachable objects from the object database
reflog              Manage reflog information
   remote              Manage set of tracked repositories
   repack              Pack unpacked objects in a repository
   replace             Create, list, delete refs to replace objects

Ancillary Commands / Interrogators
   annotate            Annotate file lines with commit information
   blame               Show what revision and author last modified each line of a file
bugreport           Collect information for user to file a bug report
   count-objects       Count unpacked number of objects and their disk consumption
difftool            Show changes using common diff tools
fsck                Verifies the connectivity and validity of the objects in the database
gitweb              Git web interface (web frontend to Git repositories)
   help                Display help information about Git
instaweb            Instantly browse your working repository in gitweb
```

```
   merge-tree          Show three-way merge without touching index
rerere              Reuse recorded resolution of conflicted merges
   show-branch         Show branches and their commits
   verify-commit       Check the GPG signature of commits
   verify-tag          Check the GPG signature of tags
whatchanged         Show logs with difference each commit introduces

Interacting with Others
archimport          Import a GNU Arch repository into Git
cvsexportcommit     Export a single commit to a CVS checkout
cvsimport           Salvage your data out of another SCM people love to hate
cvsserver           A CVS server emulator for Git
imap-send           Send a collection of patches from stdin to an IMAP folder
   p4                Import from and submit to Perforce repositories
quiltimport         Applies a quilt patchset onto the current branch
   request-pull        Generates a summary of pending changes
   send-email          Send a collection of patches as emails
svn                 Bidirectional operation between a Subversion repository and Git

Low-level Commands / Manipulators
   apply               Apply a patch to files and/or to the index
   checkout-index      Copy files from the index to the working tree
   commit-graph        Write and verify Git commit-graph files
   commit-tree         Create a new commit object
   hash-object         Compute object ID and optionally creates a blob from a file
   index-pack          Build pack index file for an existing packed archive
   merge-file          Run a three-way file merge
   merge-index         Run a merge for files needing merging
mktag               Creates a tag object
mktree              Build a tree-object from ls-tree formatted text
   multi-pack-index    Write and verify multi-pack-indexes
   pack-objects        Create a packed archive of objects
   prune-packed        Remove extra objects that are already in pack files
   read-tree           Reads tree information into the index
   symbolic-ref        Read, modify and delete symbolic refs
   unpack-objects      Unpack objects from a packed archive
   update-index        Register file contents in the working tree to the index
   update-ref          Update the object name stored in a ref safely
   write-tree          Create a tree object from the current index

Low-level Commands / Interrogators
   cat-file            Provide content or type and size information for repository objects
   cherry              Find commits yet to be applied to upstream
   diff-files          Compares files in the working tree and the index
   diff-index          Compare a tree to the working tree or index
   diff-tree           Compares the content and mode of blobs found via two tree objects
   for-each-ref        Output information on each ref
   for-each-repo       Run a Git command on a list of repositories
   get-tar-commit-id   Extract commit ID from an archive created using git-archive
   ls-files            Show information about files in the index and the working tree
   ls-remote           List references in a remote repository
   ls-tree             List the contents of a tree object
   merge-base          Find as good common ancestors as possible for a merge
   name-rev            Find symbolic names for given revs
   pack-redundant      Find redundant pack files
   rev-list            Lists commit objects in reverse chronological order
   rev-parse           Pick out and massage parameters
   show-index          Show packed archive index
   show-ref            List references in a local repository
   unpack-file         Creates a temporary file with a blob's contents
```

```
   var               Show a Git logical variable
   verify-pack       Validate packed Git archive files

Low-level Commands / Syncing Repositories
   daemon            A really simple server for Git repositories
   fetch-pack        Receive missing objects from another repository
   http-backend      Server side implementation of Git over HTTP
   send-pack         Push objects over Git protocol to another repository
   update-server-info   Update auxiliary info file to help dumb servers

Low-level Commands / Internal Helpers
   check-attr        Display gitattributes information
   check-ignore      Debug gitignore / exclude files
   check-mailmap     Show canonical names and email addresses of contacts
   check-ref-format  Ensures that a reference name is well formed
   column            Display data in columns
   credential        Retrieve and store user credentials
   credential-cache  Helper to temporarily store passwords in memory
   credential-store  Helper to store credentials on disk
fmt-merge-msg        Produce a merge commit message
   interpret-trailers   Add or parse structured information in commit messages
mailinfo             Extracts patch and authorship from a single e-mail message
mailsplit            Simple UNIX mbox splitter program
   merge-one-file    The standard helper program to use with git-merge-index
   patch-id          Compute unique ID for a patch
   sh-i18n           Git's i18n setup code for shell scripts
sh-setup             Common Git shell script setup code
stripspace           Remove unnecessary whitespace

External commands
askyesno
   credential-helper-selector
   flow
lfs
```

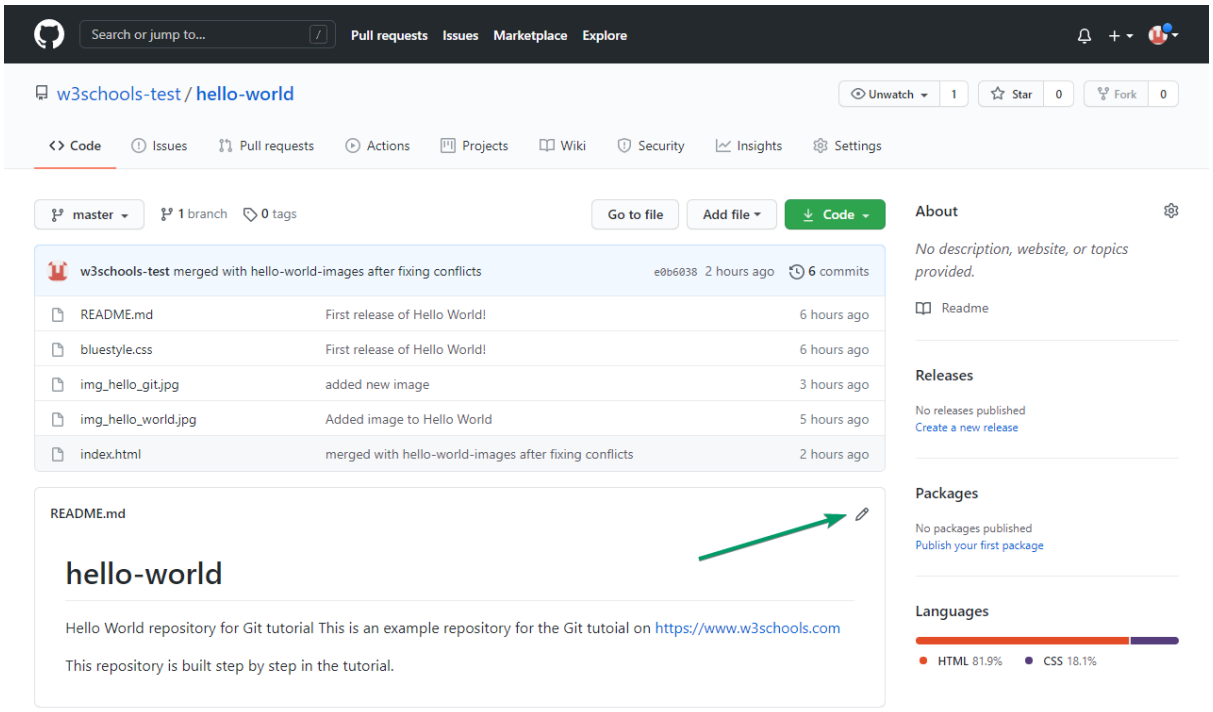
Note: If you find yourself stuck in the list view, SHIFT + G to jump the end of the list, then q to exit the view.

Git GitHub Getting Started

Edit Code in GitHub

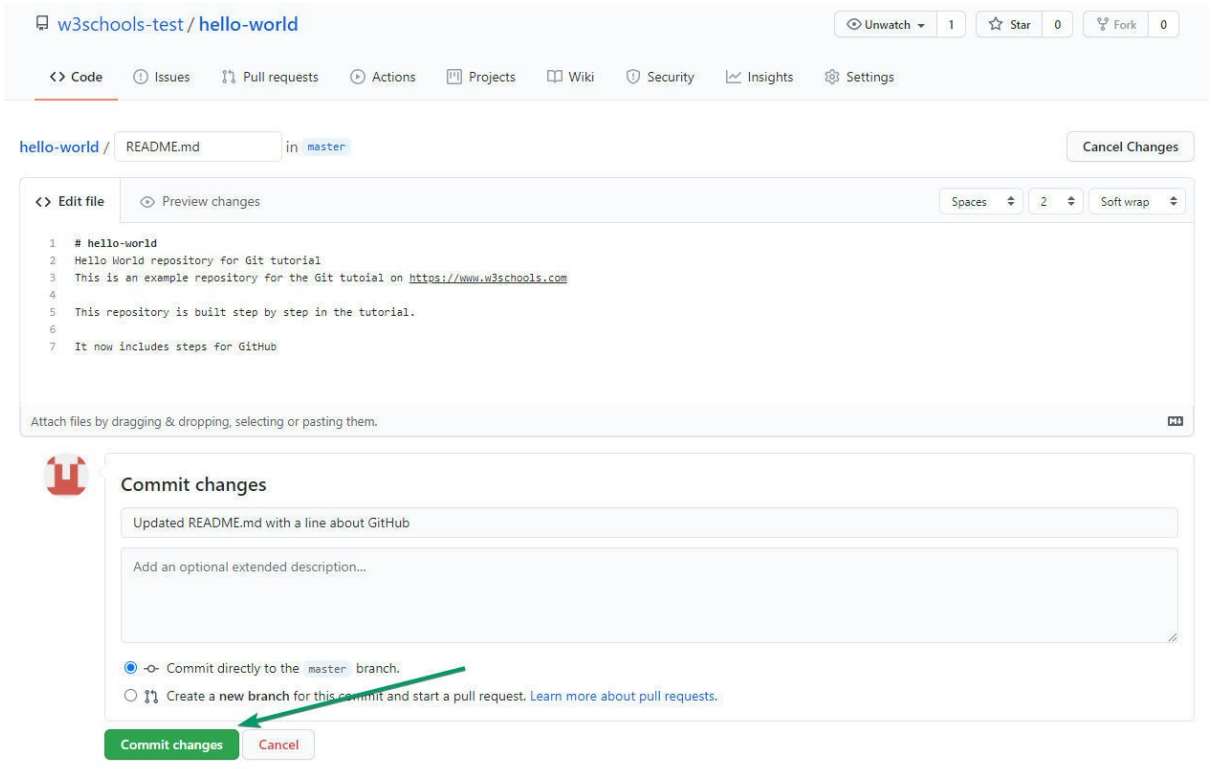
In addition to being a host for Git content, GitHub has a very good code editor.

Let's try to edit the README.md file in GitHub. Just click the edit button:



Add some changes to the code, and then commit the changes. For now, we will "Commit directly to the master branch".

Remember to add a description for the commit:



That is how you edit code directly in GitHub!

Pulling to Keep up-to-date with Changes

When working as a team on a project, it is important that everyone stays up to date.

Any time you start working on a project, you should get the most recent changes to your local copy.

With Git, you can do that with pull.

pull is a combination of 2 different commands:

- fetch
- merge

Let's take a closer look into how fetch, merge, and pull works.

Git Fetch

fetch gets all the change history of a tracked branch/repo.

So, on your local Git, fetch updates to see what has changed on GitHub:

Example

```
git fetch origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 733 bytes | 3.00 KiB/s, done.
From https://github.com/w3schools-test/hello-world
   e0b6038..d29d69f  master     -> origin/master
```

Now that we have the recent changes, we can check our status:

Example

```
git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
```

We are behind the origin/master by 1 commit. That should be the updated README.md, but lets double check by viewing the log:

Example

```
git log origin/master
commit d29d69ffe2ee9e6df6fa0d313bb0592b50f3b853 (origin/master)
```

```
Author: w3schools-test <77673807+w3schools-test@users.noreply.github.com>
Date:   Fri Mar 26 14:59:14 2021 +0100

    Updated README.md with a line about GitHub

commit e0b6038b1345e50aca8885d8fd322fc0e5765c3b (HEAD -> master)
Merge: dfa79db 1f1584e
Author: w3schools-test
Date:   Fri Mar 26 12:42:56 2021 +0100

    merged with hello-world-images after fixing conflicts

...
...
```

That looks as expected, but we can also verify by showing the differences between our local master and origin/master:

Example

```
gitdiff origin/master
diff --git a/README.md b/README.md
index 23a0122..a980c39 100644
--- a/README.md
+++ b/README.md
@@ -2,6 +2,4 @@
 Hello World repository for Git tutorial
 This is an example repository for the Git tutoial on https://www.w3schools.com

-This repository is built step by step in the tutorial.
-
-It now includes steps for GitHub
+This repository is built step by step in the tutorial.
\ No newline at end of file
```

That looks precisely as expected! Now we can safely merge.

Git Merge

merge combines the current branch, with a specified branch.

We have confirmed that the updates are as expected, and we can merge our current branch (master) with origin/master:

Example

```
git merge origin/master
Updating e0b6038..d29d69f
Fast-forward
 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Check our status again to confirm we are up to date:

Example

```
git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

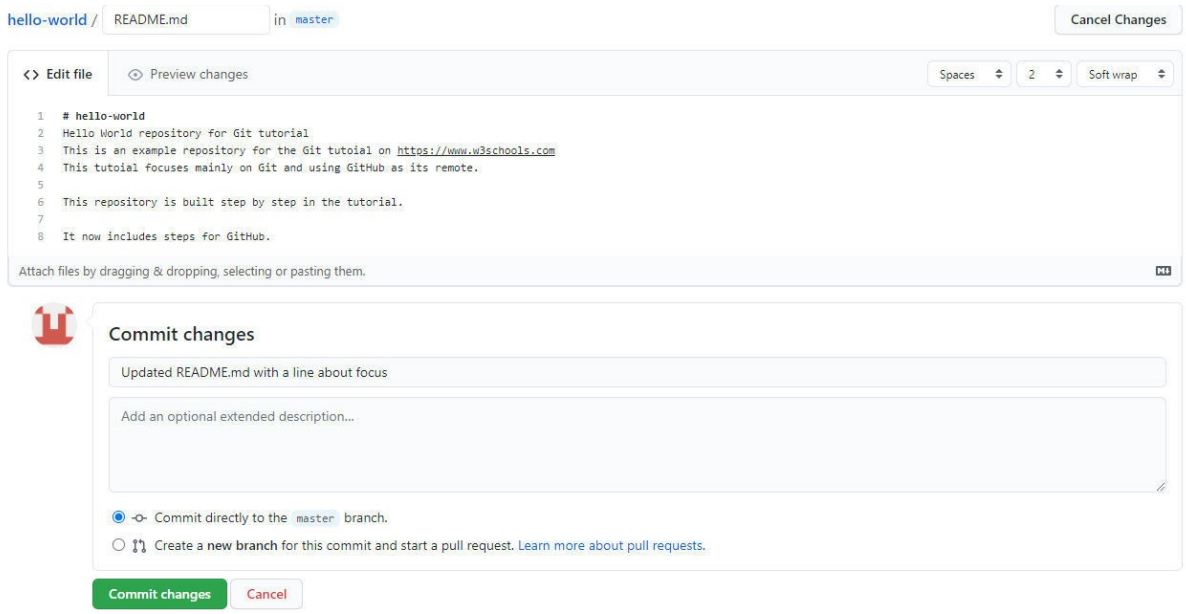
There! Your local git is up to date!

Git Pull

But what if you just want to update your local repository, without going through all those steps?

pull is a combination of fetch and merge. It is used to pull all changes from a remote repository into the branch you are working on.

Make another change to the Readme.md file on GitHub.



Use pull to update our local Git:

Example

```
git pull origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 794 bytes | 1024 bytes/s, done.
From https://github.com/w3schools-test/hello-world
   a7cdd4b..ab6b4ed  master       -> origin/master
Updating a7cdd4b..ab6b4ed
Fast-forward
```

```
README.md | 2 ++
1 file changed, 2 insertions(+)
```

That is how you keep your local Git up to date from a remote repository. In the next chapter, we will look closer at how `push` works on GitHub.

Git Push to GitHub

Push Changes to GitHub

Let's try making some changes to our local git and pushing them to GitHub.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
<link rel="stylesheet" href="bluestyle.css">
</head>
<body>

<h1>Hello world!</h1>
<div><imgsrc="img_hello_world.jpg" alt="Hello World from Space" style="width:100%;max-width:640px"></div>
<p>This is the first file in my new Git Repo.</p>
<p>This line is here to show how merging works.</p>
<div><imgsrc="img_hello_git.jpg" alt="Hello Git" style="width:100%;max-width:640px"></div>

</body>
</html>
```

Commit the changes:

Example

```
git commit -a -m "Updated index.html. Resized image"
[master e7de78f] Updated index.html. Resized image
 1 file changed, 1 insertion(+), 1 deletion(-)
```

And check the status:

Example

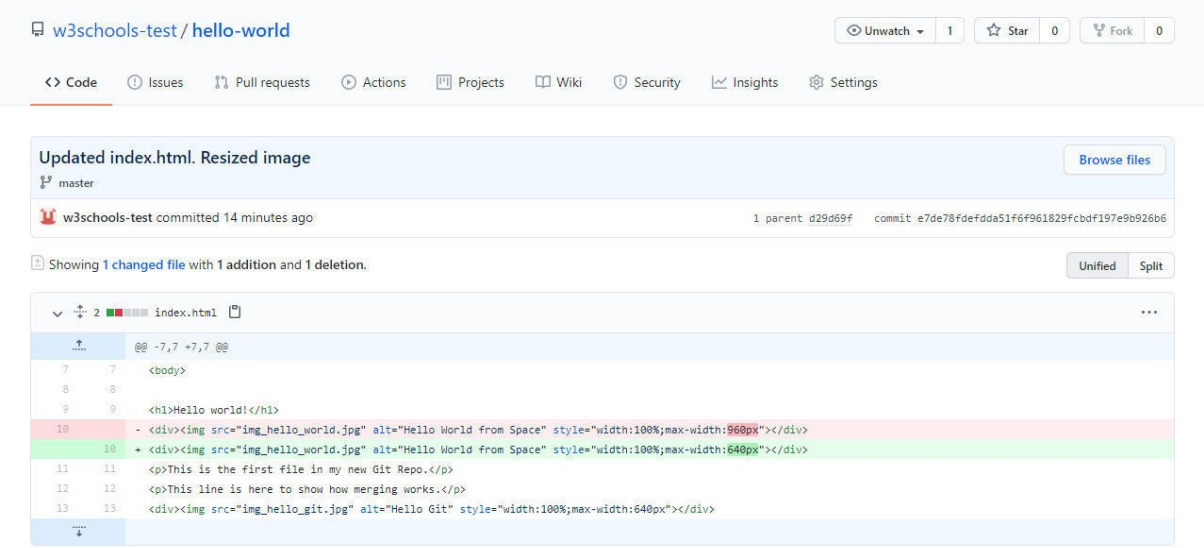
```
git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Now push our changes to our remote origin:

Example

```
git push origin
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 578 bytes | 578.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/w3schools-test/hello-world.git
5a04b6f..facaeae  master -> master
```

Go to GitHub, and confirm that the repository has a new commit:



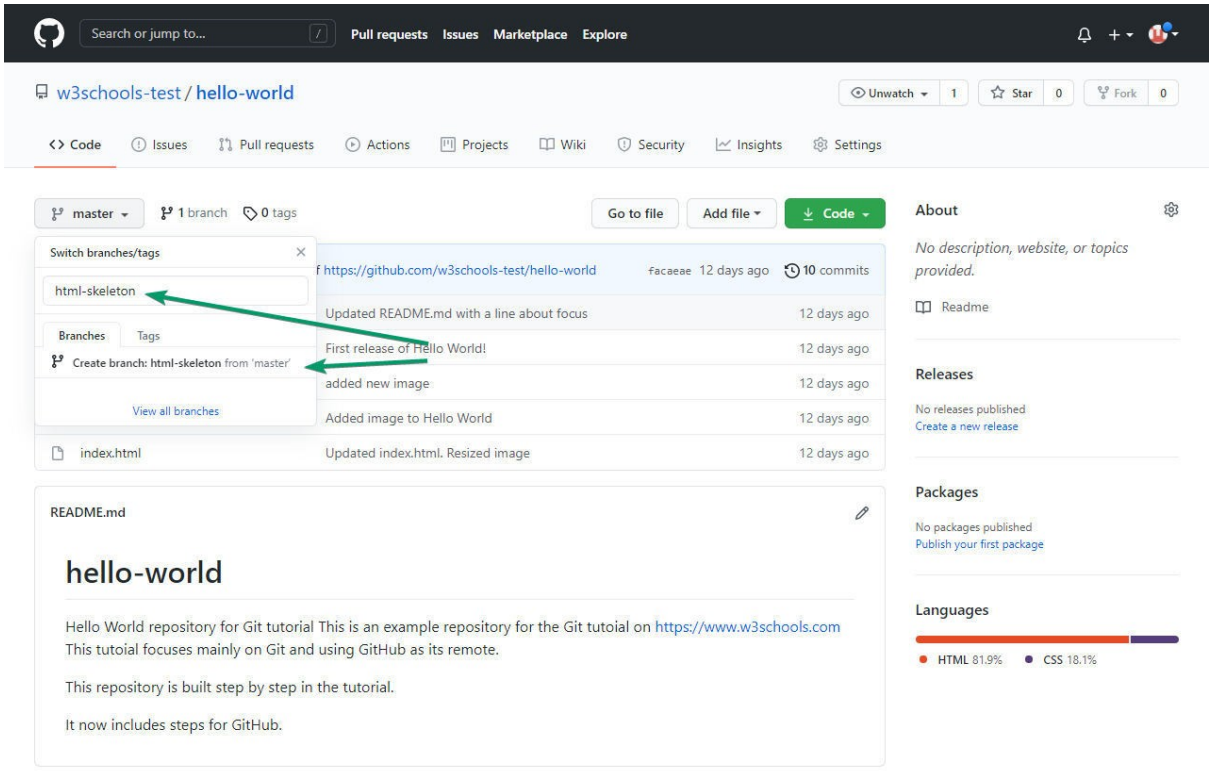
Now, we are going to start working on branches on GitHub.

Git GitHub Branch

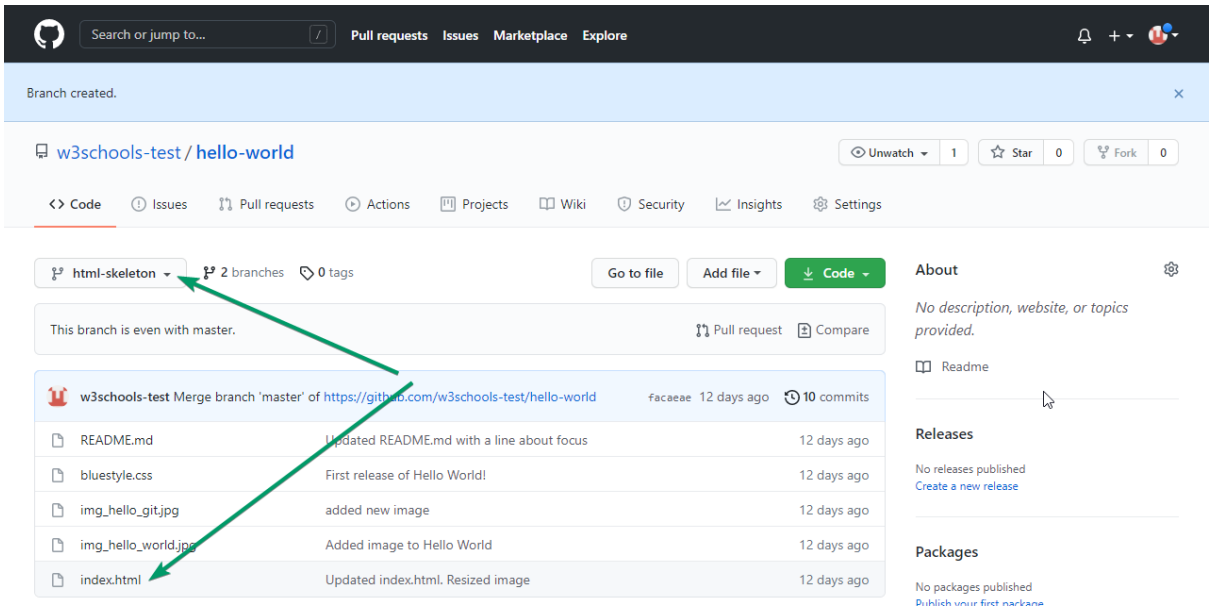
Create a New Branch on GitHub

On GitHub, access your repository and click the "master" branch button.

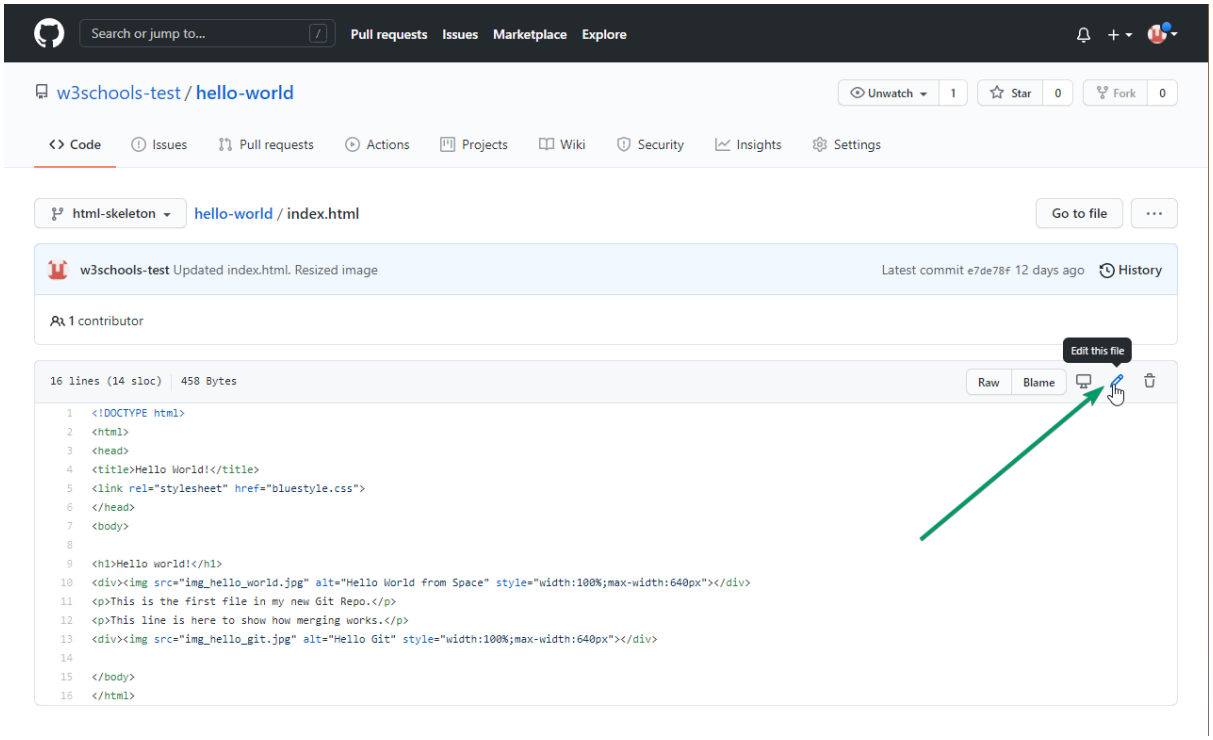
There you can create a new Branch. Type in a descriptive name, and click Create branch:



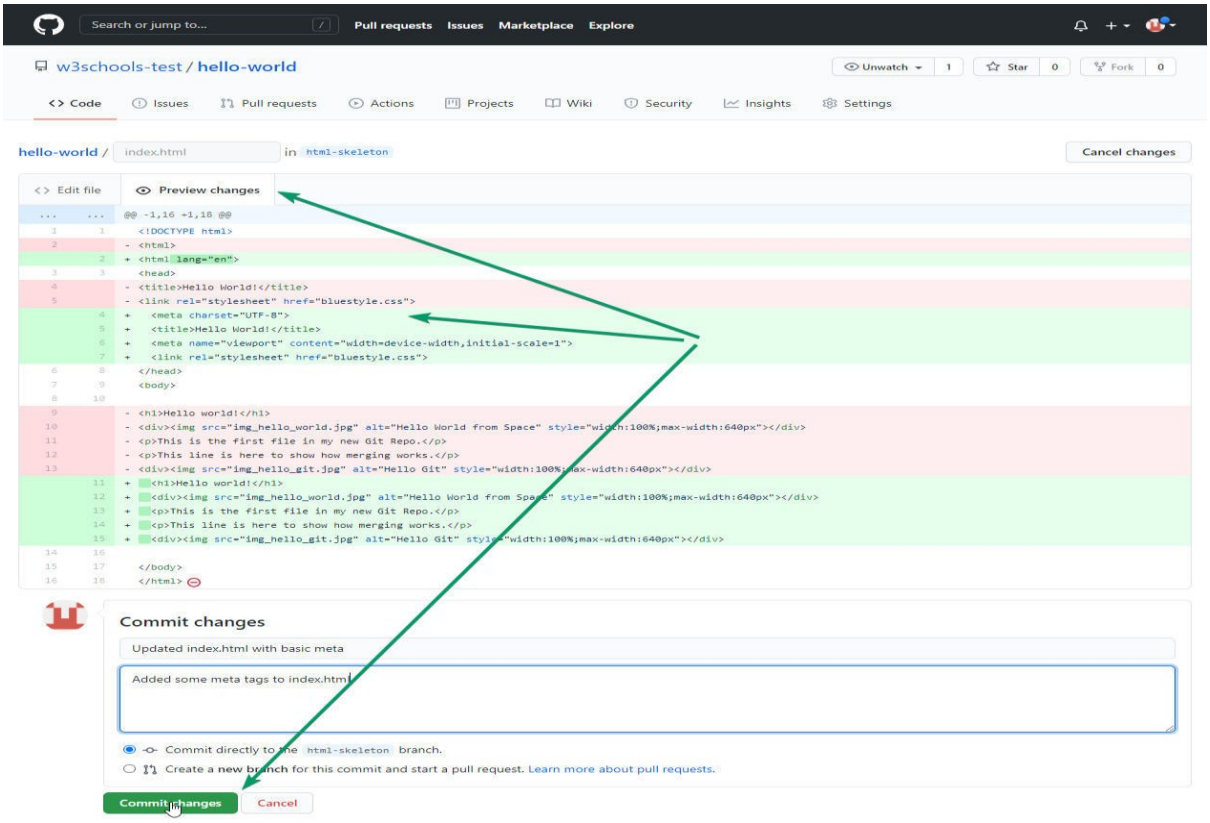
The branch should now be created and active. You can confirm which branch you are working on by looking at the branch button. See that it now says "html-skeleton" instead of "main"?



Start working on an existing file in this branch. Click the "index.html" file and start editing:



After you have finished editing the file, you can click the "Preview changes" tab to see the changes you made highlighted:



If you are happy with the change, add a comment that explains what you did, and click Commit changes.

Git Pull Branch from GitHub

Pulling a Branch from GitHub

Now continue working on our new branch in our local Git.

Letspull from our GitHub repository again so that our code is up-to-date:

Example

```
gitpull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 851 bytes | 9.00 KiB/s, done.
From https://github.com/w3schools-test/hello-world
 * [new branch]      html-skeleton -> origin/html-skeleton
Already up to date.
```

Now our main branch is up todate. And we can see that there is a new branch available on GitHub.

Do a quick status check:

Example

```
git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

And confirm which branches we have, and where we are working at the moment:

Example

```
git branch
* master
```

So, we do not have the new branch on our local Git. But we know it is available on GitHub. So we can use the -a option to see all local and remote branches:

Example

```
git branch -a
* master
  remotes/origin/html-skeleton
  remotes/origin/master
```

Note:branch -r is for remote branches only.

We see that the branch html-skeleton is available remotely, but not on our local git. Lets check it out:

Example

```
git checkout html-skeleton
Switched to a new branch 'html-skeleton'
Branch 'html-skeleton' set up to track remote branch 'html-skeleton' from 'origin'.
```

And check if it is all up to date:

Example

```
gitpull
Already up to date.
```

Which branches do we have now, and where are we working from?

Example

```
git branch
* html-skeleton
  master
```

Now, open your favourite editor and confirm that the changes from the GitHub branch carried over.

That is how you pull a GitHub branch to your local Git.

Git Push Branch to GitHub

Push a Branch to GitHub

Let's try to create a new local branch, and push that to GitHub.

Start by creating a branch, like we did earlier:

Example

```
git checkout -b update-readme
Switched to a new branch 'update-readme'
```

And we make some changes to the README.md file. Just add a new line.

So now we check the status of the current branch.

Example

```
git status
On branch update-readme
Changes not staged for commit:
  (use "git add ..." to update what will be committed)
  (use "git restore ..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

We see that README.md is modified but not added to the Staging Environment:

Example

```
gitadd README.md
```

Check the status of the branch:

Example

```
git status
On branch update-readme
Changes to be committed:
  (use "git restore --staged ..." to unstage)
        modified:   README.md
```

We are happy with our changes. So we will commit them to the branch:

Example

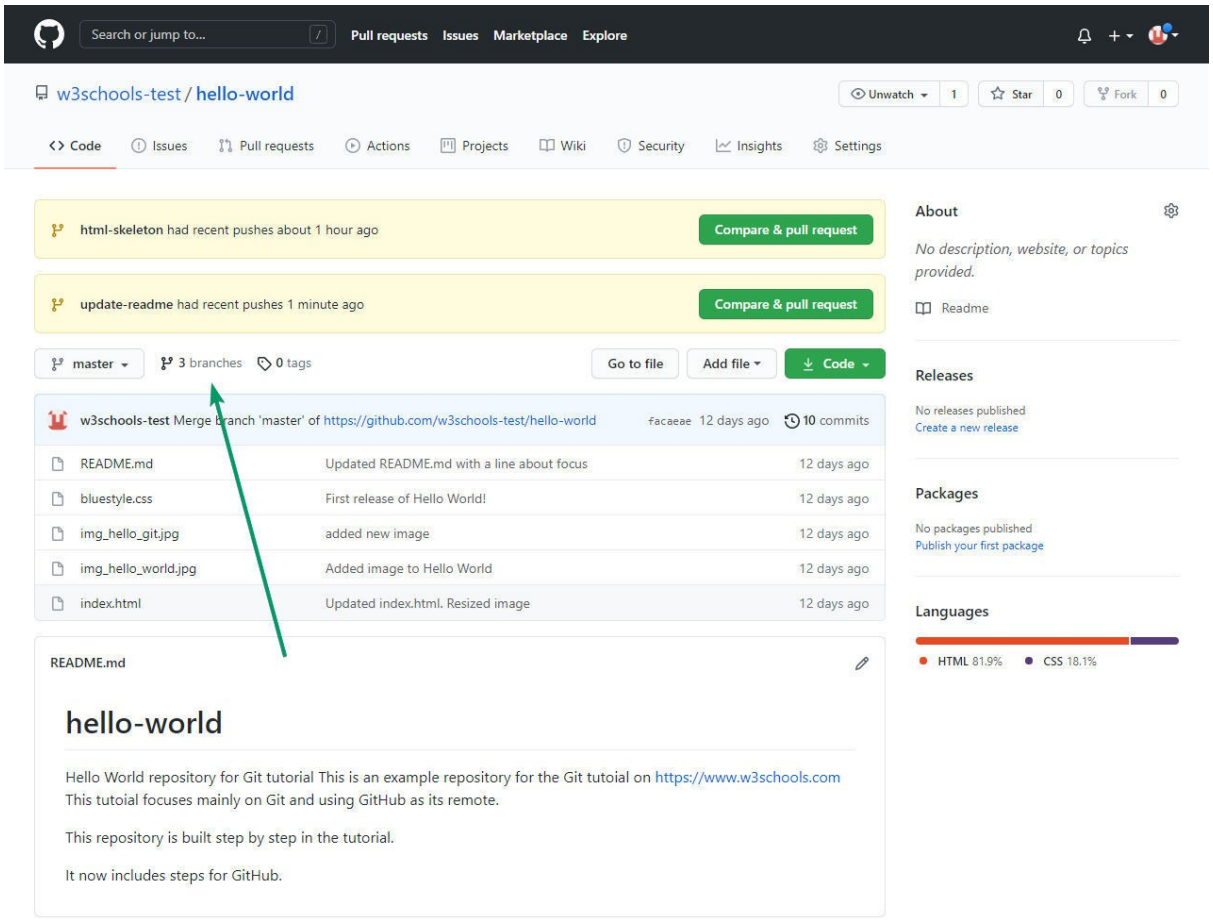
```
git commit -m "Updated readme for GitHub Branches"
[update-readme 836e5bf] Updated readme for GitHub Branches
 1 file changed, 1 insertion(+)
```

Now push the branch from our local Git repository, to GitHub, where everyone can see the changes:

Example

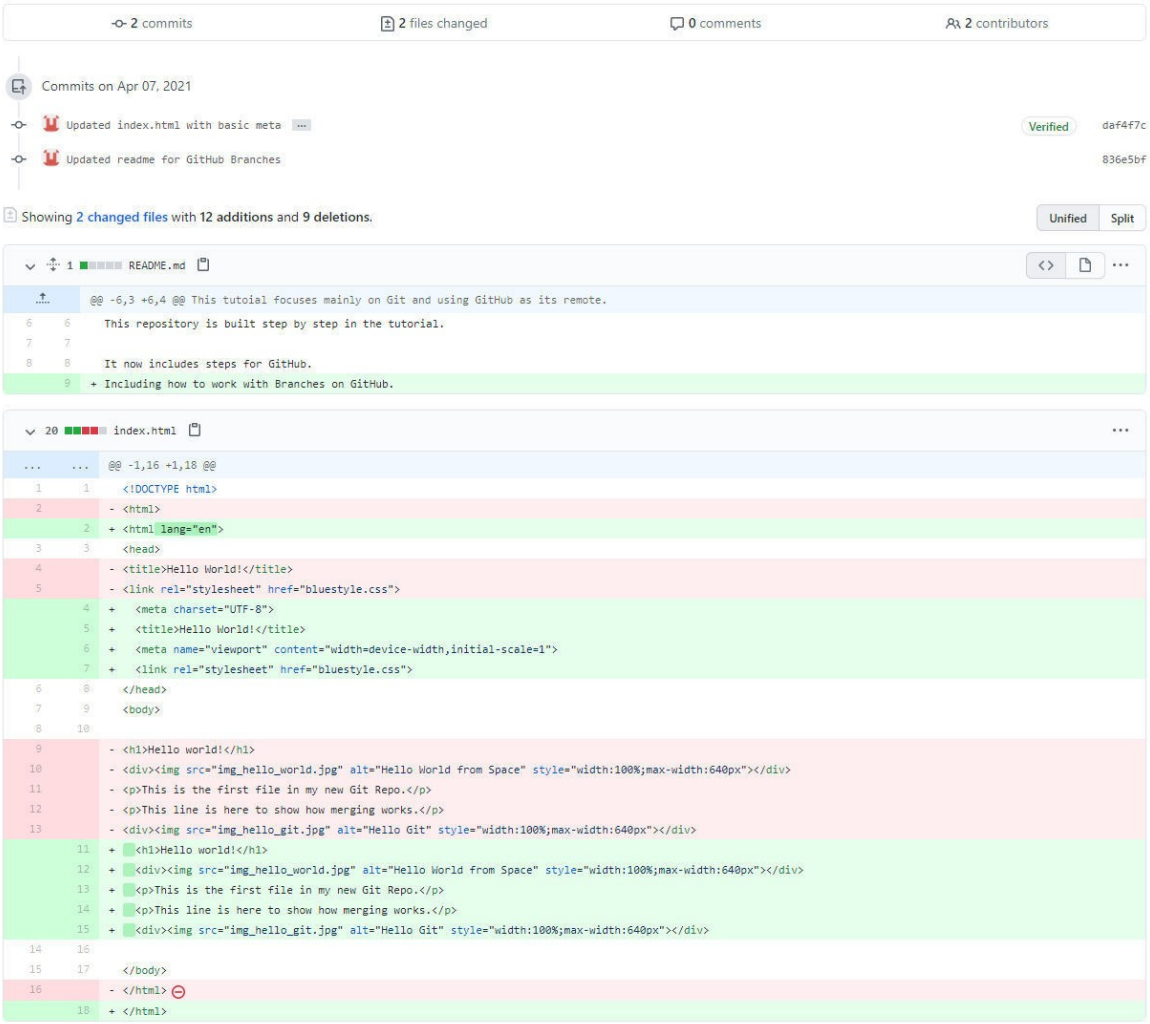
```
git push origin update-readme
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 366 bytes | 366.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'update-readme' on GitHub by visiting:
remote:      https://github.com/w3schools-test/hello-world/pull/new/update-readme
remote:
To https://github.com/w3schools-test/hello-world.git
 * [new branch]      update-readme -> update-readme
```

Go to GitHub, and confirm that the repository has a new branch:



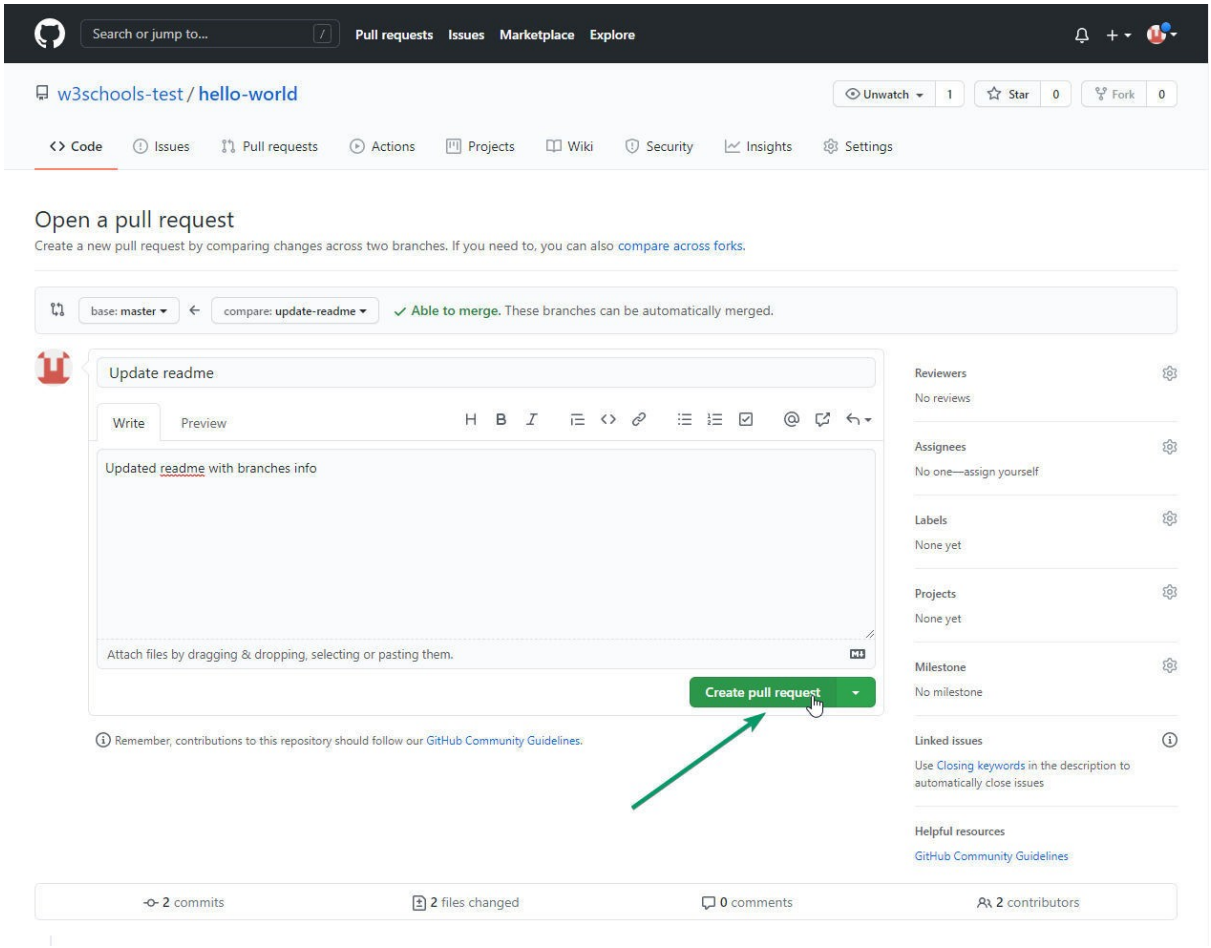
In GitHub, we can now see the changes and merge them into the master branch if we approve it.

If you click the "Compare & pull request", you can go through the changes made and new files added:



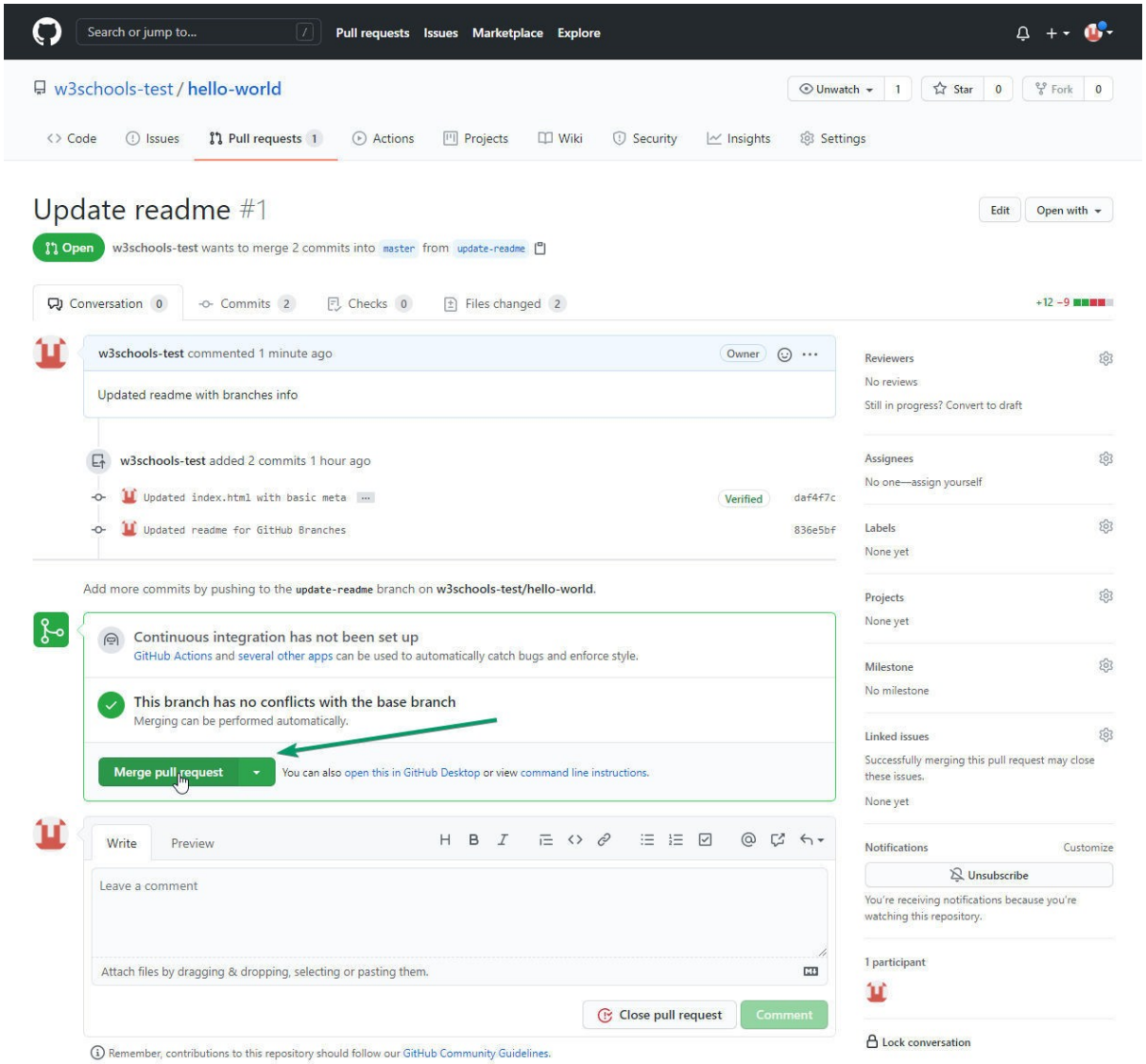
Note: This comparison shows both the changes from update-readme and html-skeleton because we created the new branch FROM html-skeleton.

If the changes look good, you can go forward, creating a pull request:



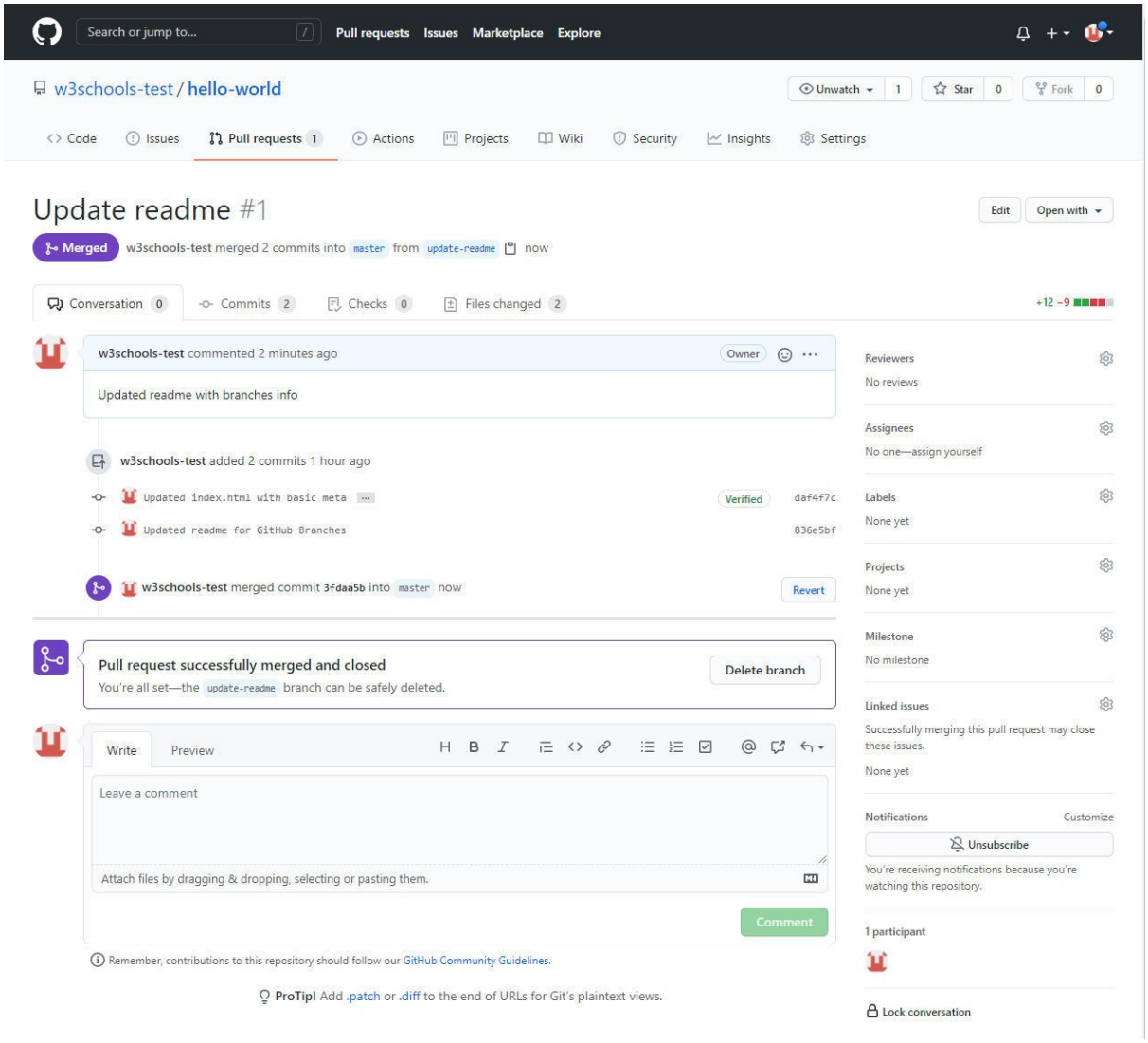
A pull request is how you propose changes. You can ask some to review your changes or pull your contribution and merge it into their branch.

Since this is your own repository, you can merge your pull request yourself:

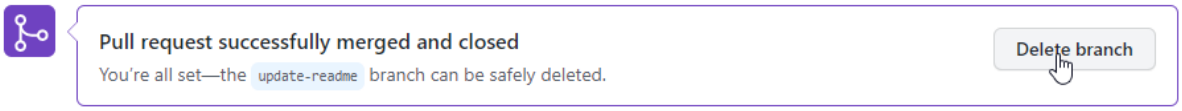


The pull request will record the changes, which means you can go through them later to figure out the changes made.

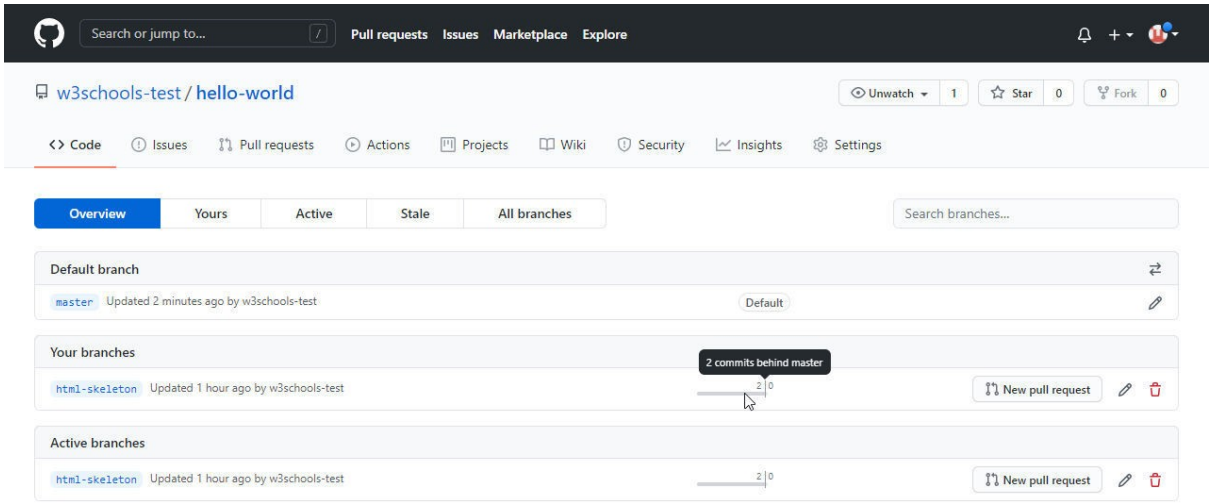
The result should be something like this:



To keep the repo from getting overly complicated, you can delete the now unused branch by clicking "Delete branch".



An after you confirm that the changes from the previous branch were included, delete that as well:



CSS Tutorial

CSS tutorial or CSS 3 tutorial provides basic and advanced concepts of CSS technology. Our CSS tutorial is developed for beginners and professionals. The major points of CSS are given below:

- CSS stands for Cascading Style Sheet.
- CSS is used to design HTML tags.
- CSS is a widely used language on the web.
- HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

CSS Example with CSS Editor

In this tutorial, you will get a lot of CSS examples, you can edit and run these examples with our online CSS editor tool.

1. <!DOCTYPE>
2. <html>
3. <head>
4. <style>
5. h1{
6. color:white;
7. background-color:red;
8. padding:5px;
9. }
10. p{
11. color:blue;
12. }
13. </style>
14. </head>

```
15. <body>
16. <h1>Write Your First CSS Example</h1>
17. <p>This is Paragraph.</p>
18. </body>
19. </html>
```

Test it Now

Output:

Write Your First CSS Example

This is Paragraph.

What is CSS

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

What does CSS do

- You can add new looks to your old HTML documents.
- You can completely change the look of your website with only a few changes in CSS code.

Why use CSS

These are the three major benefits of CSS:

1) Solves a big problem

Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem. It was a W3C recommendation.

2) Saves a lot of time

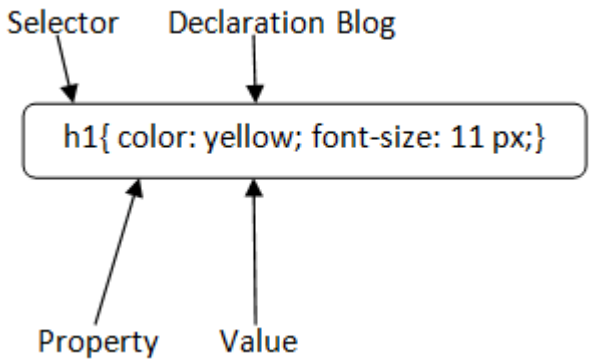
CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

3) Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

CSS Syntax

A CSS rule set contains a selector and a declaration block.



Selector: Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> etc.

Declaration Block: The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

1. color: yellow;
2. font-size: 11 px;

Each declaration contains a property name and value, separated by a colon.

Property: A Property is a type of attribute of HTML element. It could be color, border etc.

Value: Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

Selector{Property1: value1; Property2: val ue2; .....;} CSS Selector

CSS selectors are used *to select the content you want to style*. Selectors are the part of CSS

rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1. CSS Element Selector
2. CSS Id Selector
3. CSS Class Selector
4. CSS Universal Selector
5. CSS Group Selector

1) CSS Element Selector

The element selector selects the HTML element by name.

```
<!DOCTYPE html>
<html>
<head>
<style>
p{
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Test it Now

Output:

This style will be applied on every paragraph.

Me too!

And me!

---

2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

Let?s take an example with the id "para1".

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<p id="para1">Hello Javatpoint.com</p>
<p>This paragraph will not be affected.</p>
</body>
</html>
```

Test it Now

Output:

Hello Javatpoint.com

This paragraph will not be affected.

---

3) CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.

Note: A class name should not be started with a number.

Let's take an example with a class "center".

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
```

```
14. </body>
15. </html>
```

Test it Now

Output:

This heading is blue and center-aligned.

This paragraph is blue and center-aligned.

CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector.

Let's see an example.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  p.center {
6.      text-align: center;
7.      color: blue;
8.  }
9.  </style>
10. </head>
11. <body>
12. <h1 class="center">This heading is not affected</h1>
13. <p class="center">This paragraph is blue and center-aligned.</p>
14. </body>
15. </html>
```

Test it Now

Output:

This heading is not affected

This paragraph is blue and center-aligned.

---

4) CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
   color: green;
   font-size: 20px;
}
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Test it Now

Output:

This is heading

This style will be applied on every paragraph.

Me too!

And me!

---

5) CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Let's see the CSS code without group selector.

```
h1 {
    text-align: center;
    color: blue;
}
h2 {
    text-align: center;
```

```
    color: blue;
}
p {
    text-align: center;
    color: blue;
}
```

As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

```
h1,h2,p {
    text-align: center;
    color: blue;
}
```

Let's see the full example of CSS group selector.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<h1>Hello Javatpoint.com</h1>
<h2>Hello Javatpoint.com (In smaller font)</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

Output:

Hello Javatpoint.com

Hello Javatpoint.com (In smaller font)

This is a paragraph.

How to add CSS

CSS is added to HTML pages to format the document according to information in the style sheet. There are three ways to insert CSS in HTML documents.

1. Inline CSS
2. Internal CSS

3. External CSS

---

1) Inline CSS

Inline CSS is used to apply CSS on a single line or element.

For example:

```
<p style="color:blue">Hello CSS</p>
```

For more visit here: Inline CSS

---

2) Internal CSS

Internal CSS is used to apply CSS on a single document or page. It can affect all the elements of the page. It is written inside the style tag within head section of html.

For example:

```
<style>
p{color:blue}
</style>
```

For more visit here: Internal CSS

---

3) External CSS

External CSS is used to apply CSS on multiple pages or all pages. Here, we write all the CSS code in a css file. Its extension must be .css for example style.css.

For example:

```
p{color:blue}
```

You need to link this style.css file to your html pages like this:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

The link tag must be used inside head section of html.

Inline CSS

We can apply CSS in a single element by inline CSS technique.

The inline CSS is also a method to insert style sheets in HTML document. This method mitigates some advantages of style sheets so it is advised to use this method sparingly.

If you want to use inline CSS, you should use the style attribute to the relevant tag.

Syntax:

```
1. <htmltag style="cssproperty1:value; cssproperty2:value;"> </htmltag>
```

Example:

```
1. <h2 style="color:red;margin-left:40px;">Inline CSS is applied on this heading.</h2>
2. <p>This paragraph is not affected.</p>
```

Output:

Inline CSS is applied on this heading.

This paragraph is not affected.

Disadvantages of Inline CSS

- You cannot use quotations within inline CSS. If you use quotations the browser will interpret this as an end of your style value.
- These styles cannot be reused anywhere else.
- These styles are tough to be edited because they are not stored at a single place.
- It is not possible to style pseudo-codes and pseudo-classes with inline CSS.
- Inline CSS does not provide browser cache advantages.

Internal CSS

The internal style sheet is used to add a unique style for a single document. It is defined in <head> section of the HTML page inside the <style> tag.

Example:

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <style>
5. body {
```

```
    background-color: linen;
}
h1 {
    color: red;
    margin-left: 80px;
}
</style>
</head>
<body>
<h1>The internal style sheet is applied on this heading.</h1>
<p>This paragraph will not be affected.</p>
</body>
</html>
```

External CSS

The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file.

It uses the <link> tag on every pages and the <link> tag should be put inside the head section.

Example:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

The external style sheet may be written in any text editor but must be saved with a .css extension. This file should not contain HTML elements.

Let's take an example of a style sheet file named "mystyle.css".

File: mystyle.css

```
body {
    background-color: lightblue;
}
h1 {
    color: navy;
    margin-left: 20px;
}
```

Note: You should not use a space between the property value and the unit. For example: It should be margin-left:20px not margin-left:20 px.

CSS Comments

CSS comments are generally written to explain your code. It is very helpful for the users who reads your code so that they can easily understand the code.

Comments are ignored by browsers.

Comments are single or multiple lines statement and written within /*.....*/ .

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    color: blue;
    /* This is a single-line comment */
    text-align: center;
}
/* This is
a multi-line
comment */
</style>
</head>
<body>
<p>Hello Javatpoint.com</p>
<p>This statement is styled with CSS.</p>
<p>CSS comments are ignored by the browsers and not shown in the output.</p>
</body>
</html>
```

Output:

Hello Javatpoint.com

This statement is styled with CSS.

CSS comments are ignored by the browsers and not shown in the output.

CSS Background

CSS background property is used to define the background effects on element. There are 5 CSS background properties that affects the HTML elements:

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

1) CSS background-color

The background-color property is used to specify the background color of the element.

You can set the background color like this:

```
<!DOCTYPE html>
<html>
<head>
<style>
h2,p{
    background-color: #b0d4de;
}
</style>
</head>
<body>
<h2>My first CSS page.</h2>
<p>Hello Javatpoint. This is an example of CSS background-color.</p>
</body>
</html>
```

Output:

My first CSS page.

Hello Javatpoint. This is an example of CSS background-color.

---

2) CSS background-image

The background-image property is used to set an image as a background of an element. By default the image covers the entire element. You can set the background image for a page like this.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
background-image: url("paper1.gif");
margin-left:100px;
}
</style>
</head>
<body>
<h1>Hello Javatpoint.com</h1>
```

```
13. </body>
14. </html>
```

Note: The background image should be chosen according to text color. The bad combination of text and background image may be a cause of poor designed and not readable webpage.

---

3) CSS background-repeat

By default, the background-image property repeats the background image horizontally and vertically. Some images are repeated only horizontally or vertically.

The background looks better if the image repeated horizontally only.

background-repeat: repeat-x;

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  body {
6.      background-image: url("gradient_bg.png");
7.      background-repeat: repeat-x;
8.  }
9.  </style>
10. </head>
11. <body>
12. <h1>Hello Javatpoint.com</h1>
13. </body>
14. </html>
```

background-repeat: repeat-y;

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  body {
6.      background-image: url("gradient_bg.png");
7.      background-repeat: repeat-y;
8.  }
9.  </style>
10. </head>
11. <body>
12. <h1>Hello Javatpoint.com</h1>
13. </body>
14. </html>
```

4) CSS background-attachment

The background-attachment property is used to specify if the background image is fixed or scroll with the rest of the page in browser window. If you set fixed the background image then the image will not move during scrolling in the browser. Let?s take an example with fixed background image.

1. background: white url('bbb.gif');
2. background-repeat: no-repeat;
3. background-attachment: fixed;

5) CSS background-position

The background-position property is used to define the initial position of the background image. By default, the background image is placed on the top-left of the webpage.

You can set the following positions:

1. center
2. top
3. bottom
4. left
5. right

1. background: white url('good-morning.jpg');
2. background-repeat: no-repeat;
3. background-attachment: fixed;
4. background-position: center;

CSS Border

The CSS border is a shorthand property used to set the border on an element.

The CSS border properties are use to specify the style, color and size of the border of an element. The CSS border properties are given below

- border-style
- border-color
- border-width
- border-radius

1) CSS border-style

The Border style property is used to specify the border type which you want to display on the web page.

There are some border style values which are used with border-style property to define a border.

Value	Description
none	It doesn't define any border.
dotted	It is used to define a dotted border.
dashed	It is used to define a dashed border.
solid	It is used to define a solid border.
double	It defines two borders wIth the same border-width value.
groove	It defines a 3d grooved border. effect is generated according to border-color value.
ridge	It defines a 3d ridged border. effect is generated according to border-color value.
inset	It defines a 3d inset border. effect is generated according to border-color value.
outset	It defines a 3d outset border. effect is generated according to border-color value.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.none {border-style: none;}
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.hidden {border-style: hidden;}
</style>
</head>
<body>
<p class="none">No border.</p>
<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
```

```
27. <p class="hidden">A hidden border.</p>
28. </body>
29. </html>
```

Output:

No border.

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border.

A ridge border.

An inset border.

An outset border.

A hidden border.

2) CSS border-width

The border-width property is used to set the border's width. It is set in pixels. You can also use the one of the three pre-defined values, thin, medium or thick to set the width of the border.

Note: The border-width property is not used alone. It is always used with other border properties like "border-style" property to set the border first otherwise it will not work.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  p.one {
6.      border-style: solid;
7.      border-width: 5px;
8.  }
9.  p.two {
10.     border-style: solid;
11.     border-width: medium;
```

```
}
p.three {
    border-style: solid;
    border-width: 1px;
}
</style>
</head>
<body>
<p class="one">Write your text here.</p>
<p class="two">Write your text here.</p>
<p class="three">Write your text here.</p>
</body>
</html>
```

3) CSS border-color

There are three methods to set the color of the border.

- Name: It specifies the color name. For example: "red".
- RGB: It specifies the RGB value of the color. For example: "rgb(255,0,0)".
- Hex: It specifies the hex value of the color. For example: "#ff0000".

There is also a border color named "transparent". If the border color is not set it is inherited from the color property of the element.

Note: The border-color property is not used alone. It is always used with other border properties like "border-style" property to set the border first otherwise it will not work.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
    border-style: solid;
    border-color: red;
}
p.two {
    border-style: solid;
    border-color: #98bf21;
}
</style>
</head>
<body>
<p class="one">This is a solid red border</p>
<p class="two">This is a solid green border</p>
</body>
</html>
```

CSS border-collapse property

This CSS property is used to set the border of the table cells and specifies whether the table cells share the separate or common border.

This property has two main values that are **separate** and **collapse**. When it is set to the value **separate**, the distance between the cells can be defined using the **border-spacing** property. When the **border-collapse** is set to the value **collapse**, then the **inset** value of **border-style** property behaves like **groove**, and the **outset** value behaves like **ridge**.

Syntax

1. border-collapse: separate | collapse | initial | inherit;

The values of this CSS property are defined as follows.

Property Values

separate: It is the default value that separates the border of the table cell. Using this value, each cell will display its own border.

collapse: This value is used to collapse the borders into a single border. Using this, two adjacent table cells will share a border. When this value is applied, the **border-spacing** property does not affect.

initial: It sets the property to its default value.

inherit: It inherits the property from its parent element.

Now, let's understand this CSS property by using some examples. In the first example, we are using the **separate** value of the **border-collapse** property. In the second example, we are using the **collapse** value of the **border-collapse** property.

Example - Using separate value

With this value, we can use the **border-spacing** property to set the distance between the adjacent table cells.

```
<!DOCTYPE html>
<html>

<head>
<title> border-collapse property </title>
<style>
table{
border: 2px solid blue;
text-align: center;
font-size: 20px;
width: 80%;
height: 50%;
}
th{
```

```
15. border: 5px solid red;
16. background-color: yellow;
17. }
18. td{
19. border: 5px solid violet;
20. background-color: cyan;
21. }
22. #t1 {
23. border-collapse: separate;
24. }
25. </style>
26. </head>
27.
28. <body>
29.
30. <h1> The border-collapse Property </h1>
31. <h2> border-collapse: separate; </h2>
32. <table id = "t1">
33. <tr>
34. <th> First_Name </th>
35. <th> Last_Name </th>
36. <th> Subject </th>
37. <th> Marks </th>
38. </tr>
39. <tr>
40. <td> James </td>
41. <td> Gosling </td>
42. <td> Maths </td>
43. <td> 92 </td>
44. </tr>
45. <tr>
46. <td> Alan </td>
47. <td> Rickman </td>
48. <td> Maths </td>
49. <td> 89 </td>
50. </tr>
51. <tr>
52. <td> Sam </td>
53. <td> Mendes </td>
54. <td> Maths </td>
55. <td> 82 </td>
56. </tr>
57. </table>
58. </body>
59.
60. </html>
```

Test it Now

Output

Example - Using collapse property

The **border-spacing** and **border-radius** properties cannot be used with this value.

```
<!DOCTYPE html>
<html>

<head>
<title> border-collapse property </title>
<style>
table{
border: 2px solid blue;
text-align: center;
font-size: 20px;
width: 80%;
height: 50%;
}
th{
border: 5px solid red;
background-color: yellow;
}
td{
border: 5px solid violet;
background-color: cyan;
}
#t1{
border-collapse: collapse;
}
</style>
</head>

<body>

<h1> The border-collapse Property </h1>
<h2> border-collapse: collapse; </h2>
<table id = "t1">
<tr>
<th> First_Name </th>
<th> Last_Name </th>
<th> Subject </th>
<th> Marks </th>
</tr>
<tr>
<td> James </td>
<td> Gosling </td>
<td> Maths </td>
<td> 92 </td>
</tr>
<tr>
<td> Alan </td>
```

```
47. <td> Rickman </td>
48. <td> Maths </td>
49. <td> 89 </td>
50. </tr>
51. <tr>
52. <td> Sam </td>
53. <td> Mendes </td>
54. <td> Maths </td>
55. <td> 82 </td>
56. </tr>
57. </table>
58. </body>
59. </html>
```

Test it Now

Output

CSS border-spacing property

This CSS property is used to set the distance between the borders of the adjacent cells in the table. It applies only when the **border-collapse** property is set to **separate**. There will not be any space between the borders if the **border-collapse** is set to **collapse**.

It can be defined as one or two values for determining the vertical and horizontal spacing.

- When only one value is specified, then it sets both horizontal and vertical spacing.
- When we use the two-value syntax, then the first one is used to set the horizontal spacing (i.e., the space between the adjacent columns), and the second value sets the vertical spacing (i.e., the space between the adjacent rows).

Syntax

1. border-spacing: length | initial | inherit;

Property Values

The values of this CSS property are defined as follows.

length: This value sets the distance between the borders of the adjacent table cells in px, cm, pt, etc. Negative values are not allowed.

initial: It sets the property to its default value.

inherit: It inherits the property from its parent element.

Let's understand this CSS property by using some examples. In the first example, we are using the single value of the **border-spacing** property, and in the second example, we are

using two values of the **border-spacing** property.

Example

Here, we are using the single value of the **border-spacing** property. The **border-collapse** property is set to **separate**, and the value of the **border-spacing** is set to **45px**.

```
<!DOCTYPE html>
<html>

<head>
<title> border-spacing property </title>
<style>
table{
border: 2px solid blue;
text-align: center;
font-size: 20px;
background-color: lightgreen;
}
th{
border: 5px solid red;
background-color: yellow;
}
td{
border: 5px solid violet;
background-color: cyan;
}
#space{
border-collapse: separate;
border-spacing: 45px;
}
</style>
</head>

<body>

<h1> The border-spacing Property </h1>
<h2> border-spacing: 45px; </h2>
<table id = "space">
<tr>
<th> First_Name </th>
<th> Last_Name </th>
<th> Subject </th>
<th> Marks </th>
</tr>
<tr>
<td> James </td>
<td> Gosling </td>
<td> Maths </td>
<td> 92 </td>
</tr>
<tr>
```

```
<td> Alan </td>
<td> Rickman </td>
<td> Maths </td>
<td> 89 </td>
</tr>
<tr>
<td> Sam </td>
<td> Mendes </td>
<td> Maths </td>
<td> 82 </td>
</tr>
</table>
</body>

</html>
```

Output

Example

Here, we are using two values of the **border-spacing** property. The **border-collapse** property is set to **separate**, and the value of the **border-spacing** is set to **20pt 1em**. The first value, i.e., **20pt** sets the horizontal spacing, and the value **1em** set the vertical spacing.

```
<!DOCTYPE html>
<html>

<head>
<title> border-spacing property </title>
<style>
table{
border: 2px solid blue;
text-align: center;
font-size: 20px;
background-color: lightgreen;
}
th{
border: 5px solid red;
background-color: yellow;
}
td{
border: 5px solid violet;
background-color: cyan;
}
#space{
border-collapse: separate;
border-spacing: 20pt 1em;
}
</style>
</head>
```

```
27.
28. <body>
29.
30. <h1> The border-spacing Property </h1>
31. <h2> border-spacing: 20pt 1em; </h2>
32. <table id = "space">
33. <tr>
34. <th> First_Name </th>
35. <th> Last_Name </th>
36. <th> Subject </th>
37. <th> Marks </th>
38. </tr>
39. <tr>
40. <td> James </td>
41. <td> Gosling </td>
42. <td> Maths </td>
43. <td> 92 </td>
44. </tr>
45. <tr>
46. <td> Alan </td>
47. <td> Rickman </td>
48. <td> Maths </td>
49. <td> 89 </td>
50. </tr>
51. <tr>
52. <td> Sam </td>
53. <td> Mendes </td>
54. <td> Maths </td>
55. <td> 82 </td>
56. </tr>
57. </table>
58. </body>
59.
60. </html>
```

CSS Display

CSS display is the most important property of CSS which is used to control the layout of the element. It specifies how the element is displayed.

Every element has a default display value according to its nature. Every element on the webpage is a rectangular box and the CSS property defines the behavior of that rectangular box.

CSS Display default properties

default value	inline
inherited	no
animation supporting	no

version	css1
javascript syntax	object.style.display="none"

Syntax

1. display:value;

CSS display values

There are following CSS display values which are commonly used.

1. display: inline;
2. display: inline-block;
3. display: block;
4. display: run-in;
5. display: none;

---

1) CSS display inline

The inline element takes the required width only. It doesn't force the line break so the flow of text doesn't break in inline example.

The inline elements are:

-
- <a>
-
- <b> etc.

Let's see an example of CSS display inline.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <style>
5. p {
6. display: inline;
7. }
8. </style>
9. </head>
10. <body>
11. <p>Hello Javatpoint.com</p>
12. <p>Java Tutorial.</p>
13. <p>SQL Tutorial.</p>
14. <p>HTML Tutorial.</p>
15. <p>CSS Tutorial.</p>
```

```
16. </body>
17. </html>
```

Output:

Hello Javatpoint.com Java Tutorial. SQL Tutorial. HTML Tutorial. CSS Tutorial.

---

2) CSS display inline-block

The CSS display inline-block element is very similar to inline element but the difference is that you are able to set the width and height.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  p {
6.  display: inline-block;
7.  }
8.  </style>
9.  </head>
10. <body>
11. <p>Hello Javatpoint.com</p>
12. <p>Java Tutorial.</p>
13. <p>SQL Tutorial.</p>
14. <p>HTML Tutorial.</p>
15. <p>CSS Tutorial.</p>
16. </body>
17. </html>
```

Output:

Hello Javatpoint.com Java Tutorial. SQL Tutorial. HTML Tutorial. CSS Tutorial.

---

3) CSS display block

The CSS display block element takes as much as horizontal space as they can. Means the block element takes the full available width. They make a line break before and after them.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  p {
6.  display: block;
```

```
}
</style>
</head>
<body>
<p>Hello Javatpoint.com</p>
<p>Java Tutorial.</p>
<p>SQL Tutorial.</p>
<p>HTML Tutorial.</p>
<p>CSS Tutorial.</p>
</body>
</html>
```

Output:

Hello Javatpoint.com

Java Tutorial.

SQL Tutorial.

HTML Tutorial.

CSS Tutorial.

---

4) CSS display run-in

This property doesn't work in Mozilla Firefox. These elements don't produce a specific box by themselves.

- If the run-in box contains a bock box, it will be same as block.
- If the block box follows the run-in box, the run-in box becomes the first inline box of the block box.
- If the inline box follows the run-in box, the run-in box becomes a block box.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
display: run-in;
}
</style>
</head>
<body>
<p>Hello Javatpoint.com</p>
<p>Java Tutorial.</p>
<p>SQL Tutorial.</p>
```

```
14. <p>HTML Tutorial.</p>
15. <p>CSS Tutorial.</p>
16. </body>
17. </html>
```

Output:

Hello Javatpoint.com

Java Tutorial.

SQL Tutorial.

HTML Tutorial.

CSS Tutorial.

---

5) CSS display none

The "none" value totally removes the element from the page. It will not take any space.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  h1.hidden {
6.      display: none;
7.  }
8.  </style>
9.  </head>
10. <body>
11. <h1>This heading is visible.</h1>
12. <h1 class="hidden">This is not visible.</h1>
13. <p>You can see that the hidden heading does not contain any space.</p>
14. </body>
15. </html>
```

Output:

This heading is visible.

You can see that the hidden heading does not contain any space.

Other CSS display values

Property-value	Description
flex	It is used to display an element as an block-level flex container. It is new in css3.
inline-flex	It is used to display an element as an inline-level flex container. It is new in css3.
inline-table	It displays an element as an inline-level table.
list-Item	It makes the element behave like a element.
table	It makes the element behave like a <table> element.
table-caption	It makes the element behave like a <caption> element.
table-column-group	It makes the element behave like a <colgroup> element.
table-header-group	It makes the element behave like a <thead> element.
table-footer-group	It makes the element behave like a <tfoot> element.
table-row-group	It makes the element behave like a <tbody> element.
table-cell	It makes the element behave like a <td> element.
table-row	It makes the element behave like a <tr> element.
table-column	It makes the element behave like a <col> element.

CSS Cursor

It is used to define the type of mouse cursor when the mouse pointer is on the element. It allows us to specify the cursor type, which will be displayed to the user. When a user hovers on the link, then by default, the cursor transforms into the hand from a pointer.

Let's understand the property values of the cursor.

Values	Usage
alias	It is used to display the indication of the cursor of something that is to be created.
auto	It is the default property in which the browser sets the cursor.
all-scroll	It indicates the scrolling.
col-resize	Using it, the cursor will represent that the column can be horizontally resized.
cell	The cursor will represent that a cell or the collection of cells is selected.
context-menu	It indicates the availability of the context menu.
default	It indicates an arrow, which is the default cursor.
copy	It is used to indicate that something is copied.
crosshair	In it, the cursor changes to the crosshair or the plus sign.
e-resize	It represents the east direction and indicates that the edge of the box is to be shifted towards right.
ew-resize	It represents the east/west direction and indicates a bidirectional resize cursor.
n-resize	It represents the north direction that indicates that the edge of the box is to be

	shifted to up.
ne-resize	It represents the north/east direction and indicates that the edge of the box is to be shifted towards up and right.
move	It indicates that something is to be shifted.
help	It is in the form of a question mark or ballon, which represents that help is available.
None	It is used to indicate that no cursor is rendered for the element.
No-drop	It is used to represent that the dragged item cannot be dropped here.
s-resize	It indicates an edge box is to be moved down. It indicates the south direction.
Row-resize	It is used to indicate that the row can be vertically resized.
Se-resize	It represents the south/east direction, which indicates that an edge box is to be moved down and right.
Sw-resize	It represents south/west direction and indicates that an edge of the box is to be shifted towards down and left.
Wait	It represents an hourglass.
<url>	It indicates the source of the cursor image file.
w-resize	It indicates the west direction and represents that the edge of the box is to be shifted left.
Zoom-in	It is used to indicate that something can be zoomed in.
Zoom-out	It is used to indicate that something can be zoomed out.

The illustration of using the above values of cursor property is given below:

Example

```
<html>
  <head>
  </head>
  <style>
    body{
      background-color: lightblue;
      color:green;
      text-align: center;
      font-size: 20px;
    }

</style>

<body>
  <p>Move your mouse over the below words for the cursor change.</p>
  <div style = "cursor:alias">alias Value</div>
  <div style = "cursor:auto">auto Value</div>
  <div style = "cursor:all-scroll">all-scroll value</div>
  <div style = "cursor:col-resize">col-resize value</div>
  <div style = "cursor:crosshair">Crosshair</div>
  <div style = "cursor:default">Default value</div>
```

```
    <div style = "cursor:copy">copy value</div>
    <div style = "cursor:pointer">Pointer</div>
    <div style = "cursor:move">Move</div>
    <div style = "cursor:e-resize">e-resize</div>
    <div style = "cursor:ew-resize">ew-resize</div>
    <div style = "cursor:ne-resize">ne-resize</div>
    <div style = "cursor:nw-resize">nw-resize</div>
    <div style = "cursor:n-resize">n-resize</div>
    <div style = "cursor:se-resize">se-resize</div>
    <div style = "cursor:sw-resize">sw-resize</div>
    <div style = "cursor:s-resize">s-resize</div>
    <div style = "cursor:w-resize">w-resize</div>
    <div style = "cursor:text">text</div>
    <div style = "cursor:wait">wait</div>
    <div style = "cursor:help">help</div>
    <div style = "cursor:progress">Progress</div>
    <div style = "cursor:no-drop">no-drop</div>
    <div style = "cursor:not-allowed">not-allowed</div>
    <div style = "cursor:vertical-text">vertical-text</div>
    <div style = "cursor:zoom-in">Zoom-in</div>
    <div style = "cursor:zoom-out">Zoom-out</div>
  </body>
</html>
```

CSS Buttons

In HTML, we use the button tag to create a button, but by using CSS properties, we can style the buttons. Buttons help us to create user interaction and event processing. They are one of the widely used elements of web pages.

During the form submission, to view or to get some information, we generally use buttons.

Let's see the basic styling in buttons.

Basic styling in Buttons

There are multiple properties available that are used to style the button element. Let's discuss them one by one.

background-color

As we have discussed earlier, this property is used for setting the background color of the button element.

Syntax

```
element {
    // background-color style
```

```
}
```

Example

```
<!DOCTYPE html>
<html>

<head>
    <title>
        button background Color
    </title>

    <style>
    body{
       text-align: center;
    }
        button {
           color:lightgoldenrodyellow;
            font-size: 30px;
        }
        .b1 {
            background-color: red;
        }
        .b2 {
            background-color: blue;
        }
        .b3 {
            background-color: violet;
        }
    </style>
</head>

<body>
   <h1>The background-color property.</h1>
    <button class="b1">Red color button</button>
    <button class="b2">Blue color button</button>
    <button class="b3">Violet color button</button>
</body>
</html>
```

border

It is used to set the border of the button. It is the shorthand property for **border-width, border-color,** and **border-style**.

Syntax

```
element {
    // border style
```

```
}
```

Example

```
<!DOCTYPE html>
<html>

<head>
    <title>
        button background Color
    </title>

    <style>
    body{
      text-align: center;
    }
        button {
          color:lightgoldenrodyellow;
           font-size: 30px;
        }
        .b1 {
            background-color: red;
            border:none;
        }
        .b2 {
            background-color: blue;
            border:5px brown solid;
        }
        .b3 {
            background-color: yellow;
            color:black;
            border:5px red groove;
        }
        .b4{
          background-color:orange;
          border: 5px red dashed;
        }
        .b5{
          background-color: gray;
          border: 5px black dotted;
        }
        .b6{
          background-color: lightblue;
          border:5px blue double;
        }
    </style>
</head>

<body>
   <h1>The border property</h1>
```

```
    <button class="b1">none</button>
    <button class="b2">solid</button>
    <button class="b3">groove</button>
    <button class="b4">dashed</button>
    <button class="b5">dotted</button>
    <button class="b6">double</button>

</body>
</html>
```

border-radius

It is used to make the rounded corners of the button. It sets the border radius of the button.

Syntax

```
element {
    // border-radius property
}
```

Example

```
<!DOCTYPE html>
<html>

<head>
    <title>
        button background Color
    </title>

    <style>
    body{
      text-align: center;
    }
        button {
          color:lightgoldenrodyellow;
           font-size: 30px;
        }
        .b1 {
            background-color: red;
            border:none;
        }
        .b2 {
            background-color: blue;
            border:5px brown solid;
            border-radius: 7px;
        }
        .b3 {
            background-color: yellow;
```

```
            color:black;
            border:5px red groove;
            border-radius: 10px;
        }
        .b4{
            background-color:orange;
            border: 5px red dashed;
            border-radius: 20px;
        }
        .b5{
            background-color: gray;
            border: 5px black dotted;
            border-radius: 30px;
        }
        .b6{
            background-color: lightblue;
            border:5px blue double;
            border-radius: 25px;
        }
    </style>
</head>

<body>
    <h1>The border-radius property</h1>
    <h2>Below there is the border name and border-radius</h2>
     <button class="b1">none</button>
     <button class="b2">solid 7px</button>
     <button class="b3">groove 10px</button>
     <button class="b4">dashed 20px</button>
     <button class="b5">dotted 30px</button>
     <button class="b6">double 25px</button>

</body>
</html>
```

box-shadow

As its name implies, it is used to create the shadow of the button box. It is used to add the shadow to the button. We can also create a shadow during the hover on the button.

Syntax

```
box-shadow: [horizontal offset] [vertical offset] [blur radius]
            [optional spread radius] [color];
```

Example

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

```
3.
4.  <head>
5.      <title>
6.          button background Color
7.      </title>
8.
9.      <style>
10.     body{
11.       text-align: center;
12.     }
13.         button {
14.           color:lightgoldenrodyellow;
15.            font-size: 30px;
16.         }
17.         .b1{
18.           background-color: lightblue;
19.           border:5px red double;
20.           border-radius: 25px;
21.           color:black;
22.           box-shadow : 0 8px 16px 0 black,
23.                   0 6px 20px  0 rgba(0, 0, 0, 0.19);
24.         }
25.         .b2{
26.          background-color: lightblue;
27.           border:5px red dotted;
28.           color:black;
29.           border-radius: 25px;
30.         }
31.         .b2:hover{
32.           box-shadow : 0 8px 16px 0 black,
33.                   0 6px 20px  0 rgba(0, 0, 0, 0.19);
34.         }
35.         </style>
36. </head>
37.
38. <body>
39.     <button class="b1">Shadow on button</button>
40.     <button class="b2">Box-shadow on hover</button>
41. </body>
42. </html>
```

padding

It is used to set the button padding.

Syntax

```
1.  element {
2.      // padding style
```

```
}
```

Let's understand it using an illustration.

Example

```
<!DOCTYPE html>
<html>

<head>
    <title>
        button background Color
    </title>

    <style>
    body{
      text-align: center;
    }
       button {
          color:lightgoldenrodyellow;
           font-size: 30px;
       }
       .b1 {
           background-color: red;
           border:none;
           padding: 16px;
       }
       .b2 {
           background-color: blue;
           border:5px brown solid;
           padding:15px 30px 25px 40px;
       }
       .b3 {
           background-color: yellow;
           color:black;
           border:5px red groove;
           padding-top:30px;
       }
       .b4{
          background-color:orange;
          border: 5px red dashed;
          padding-bottom:40px;
       }
       .b5{
          background-color: gray;
          border: 5px black dotted;
          padding-left: 40px;
       }
       .b6{
          background-color: lightblue;
```

```
            border:5px blue double;
            padding-right: 40px;;
        }
    </style>
</head>

<body>
    <h1>The padding property</h1>
    <button class="b1">none</button>
    <button class="b2">solid</button>
    <button class="b3">groove</button>
    <button class="b4">dashed</button>
    <button class="b5">dotted</button>
    <button class="b6">double</button>

</body>
</html>
```

CSS Line Height

The **CSS line height property** is used *to define the minimal height of line boxes within the element*. It sets the differences between two lines of your content.

It defines the amount of space above and below inline elements. It allows you to set the height of a line of independently from the font size.

CSS line-height values

There are some property values which are used with CSS line-height property.

value	description
normal	This is a default value. it specifies a normal line height.
number	It specifies a number that is multiplied with the current font size to set the line height.
length	It is used to set the line height in px, pt,cm,etc.
%	It specifies the line height in percent of the current font.
initial	It sets this property to its default value.
inherit	It inherits this property from its parent element.

CSS line-height example

```
<!DOCTYPE html>
<html>
<head>
<style>
h3.small {
```

```
6.      line-height: 70%;
7.   }
8.   h3.big {
9.      line-height: 200%;
10. }
11. </style>
12. </head>
13. <body>
14. <h3>
15. This is a heading with a standard line-height.<br>
16. This is a heading with a standard line-height.<br>
17. The default line height in most browsers is about 110% to 120%.<br>
18. </h3>
19. <h3 class="small">
20. This is a heading with a smaller line-height.<br>
21. This is a heading with a smaller line-height.<br>
22. This is a heading with a smaller line-height.<br>
23. This is a heading with a smaller line-height.<br>
24. </h3>
25. <h3 class="big">
26. This is a heading with a bigger line-height.<br>
27. This is a heading with a bigger line-height.<br>
28. This is a heading with a bigger line-height.<br>
29. This is a heading with a bigger line-height.<br>
30. </h3>
31. </body>
32. </html>
```

CSS Margin

CSS Margin property is used to define the space around elements. It is completely transparent and doesn't have any background color. It clears an area around the element.

Top, bottom, left and right margin can be changed independently using separate properties. You can also change all properties at once by using shorthand margin property.

There are following CSS margin properties:

CSS Margin Properties

Property	Description
margin	This property is used to set all the properties in one declaration.
margin-left	it is used to set left margin of an element.
margin-right	It is used to set right margin of an element.

margin-top	It is used to set top margin of an element.
margin-bottom	It is used to set bottom margin of an element.

CSS Margin Values

These are some possible values for margin property.

Value	Description
auto	This is used to let the browser calculate a margin.
length	It is used to specify a margin pt, px, cm, etc. its default value is 0px.
%	It is used to define a margin in percent of the width of containing element.
inherit	It is used to inherit margin from parent element.

Note: You can also use negative values to overlap content.

CSS margin Example

You can define different margin for different sides for an element.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  p {
6.      background-color: pink;
7.  }
8.  p.ex {
9.       margin-top: 50px;
10.     margin-bottom: 50px;
11.     margin-right: 100px;
12.     margin-left: 100px;
13. }
14. </style>
15. </head>
16. <body>
17. <p>This paragraph is not displayed with specified margin. </p>
18. <p class="ex">This paragraph is displayed with specified margin.</p>
19. </body>
20. </html>
```

Output:

This paragraph is not displayed with specified margin.

This paragraph is displayed with specified margin.

Margin: Shorthand Property

CSS shorthand property is used to shorten the code. It specifies all the margin properties in one property.

There are four types to specify the margin property. You can use one of them.

1. margin: 50px 100px 150px 200px;
2. margin: 50px 100px 150px;
3. margin: 50px 100px;
4. margin 50px;

---

1) margin: 50px 100px 150px 200px;

It identifies that:

top margin value is 50px

right margin value is 100px

bottom margin value is 150px

left margin value is 200px

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    background-color: pink;
}
p.ex {
     margin: 50px 100px 150px 200px;
}
</style>
</head>
<body>
<p>This paragraph is not displayed with specified margin. </p>
<p class="ex">This paragraph is displayed with specified margin.</p>
</body>
```

```
17. </html>
```

Output:

This paragraph is not displayed with specified margin.

This paragraph is displayed with specified margin.

---

2) margin: 50px 100px 150px;

It identifies that:

top margin value is 50px

left and right margin values are 100px

bottom margin value is 150px

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  p {
6.      background-color: pink;
7.  }
8.  p.ex {
9.      margin: 50px 100px 150px;
10. }
11. </style>
12. </head>
13. <body>
14. <p>This paragraph is not displayed with specified margin. </p>
15. <p class="ex">This paragraph is displayed with specified margin.</p>
16. </body>
17. </html>
```

Output:

This paragraph is not displayed with specified margin.

This paragraph is displayed with specified margin.

---

3) margin: 50px 100px;

It identifies that:

top and bottom margin values are 50px

left and right margin values are 100px

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    background-color: pink;
}
p.ex {
     margin: 50px 100px;
}
</style>
</head>
<body>
<p>This paragraph is not displayed with specified margin. </p>
<p class="ex">This paragraph is displayed with specified margin.</p>
</body>
</html>
```

Output:

This paragraph is not displayed with specified margin.

This paragraph is displayed with specified margin.