HTML – A Beginners Guide

Topic 1: Introduction to HTML

HTML is an acronym for **HyperText Markup Language**. Unlike a scripting or programming language, that uses scripts to perform functions, a markup language uses tags to identify content.

Here is an example of an HTML tag:

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

In this example, you can see angular brackets enclosing the character "p" (<p>), and another pair of angular brackets enclosing "p" preceded by a forward slash (</p>). These are nothing, but the opening and closing tags used in HTML. The important thing to note here is that, character "p" denotes a "paragraph" here, and whatever is written within the opening (<p>), and closing (</p>) paragraph tags, will be displayed in the browser.

Topic 2: The Web Structure

The ability to code using HTML is essential for any web professional. Acquiring this skill should be the starting point for anyone who is learning how to create content for the web. The **Modern Web Design** includes 3 layers, which are as follows:

1. HTML: It provides an initial structure to the web page.
2. CSS: It is responsible for adding presentation to that web page. In other words, it defines the view of the page.
3. JavaScript: This adds behaviour to the web page, for example, it makes the page to interact with the end user.

For backend, we need to have a technology, for example, PHP, Java, ASP.Net, Ruby, or Node.js.

For managing the content, we can make use of the Content Management Systems like Wordpress, Joomla, Drupal, Magento, Squarespace, Wix, Ghost, etc.

Topic 3: HTML Basic Tags

The <html> tag

Although various versions have been released over the years, HTML basics remain the same.

The structure of an HTML document has been compared with that of a sandwich. As a sandwich has two slices of bread, the HTML document has opening and closing HTML tags.

These tags, like the bread in a sandwich, surround everything else:

```
<html>

…

</html>
```

Everything in an HTML document is surrounded by the <html> tag.

The <head> Tag

Immediately following the opening HTML tag, you'll find the head of the document, which is identified by opening and closing head tags.

The head of an HTML file contains all of the non-visual elements that help make the page work.

```
<html>

<head>…</head>

</html>
```

The head section elements will be discussed later.

The <body> Tag

The body tag follows the head tag. All visual-structural elements are contained within the body tag.

Headings, paragraphs, lists, quotes, images, and links are just a few of the elements that can be contained within the body tag.

Basic HTML Structure:

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

The <body> tag defines the main content of the HTML document.

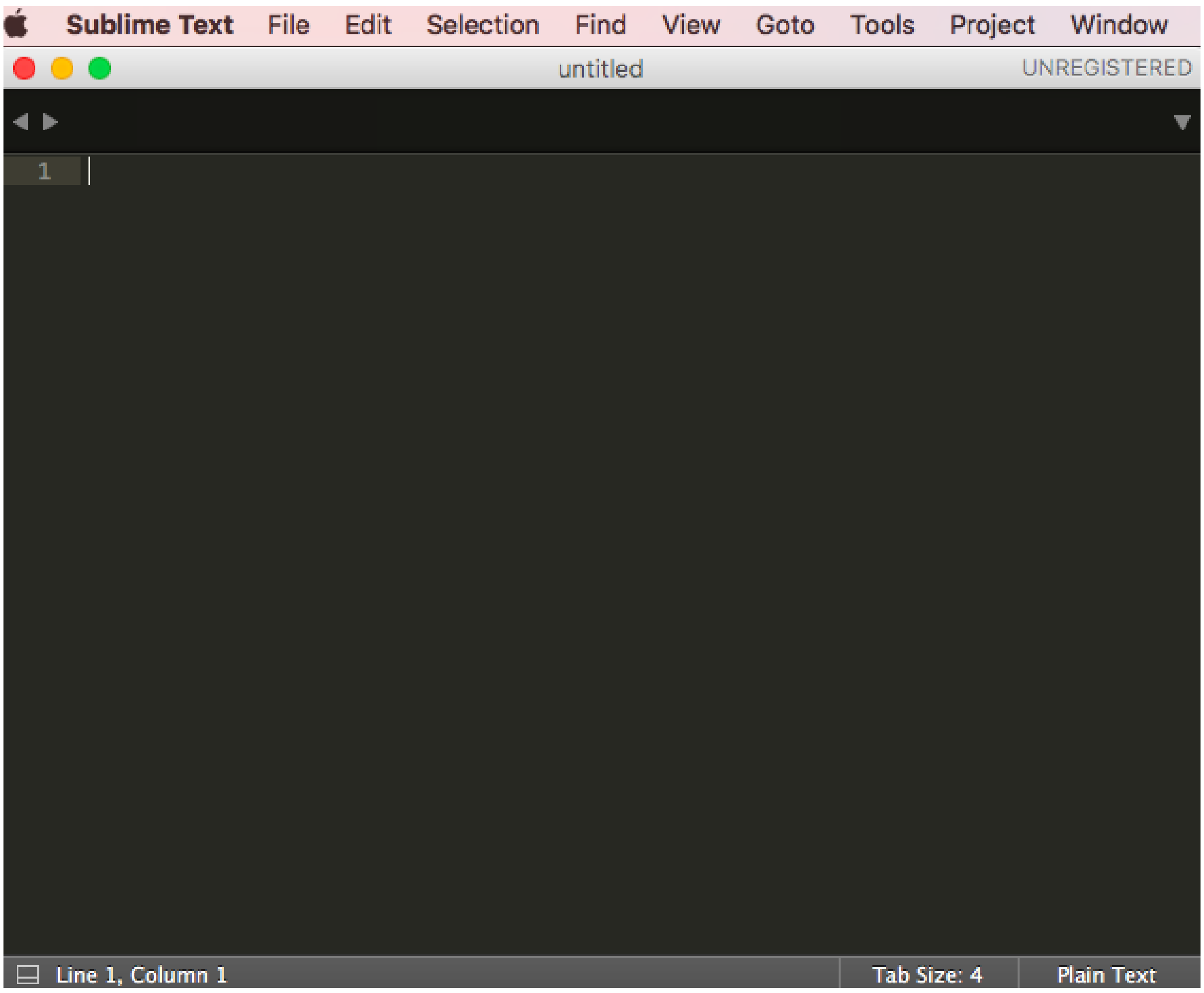
The HTML File

HTML files are text files, so you can use any text editor to create your first webpage.

There are some very nice HTML editors available. Some examples of the text editors are listed below.

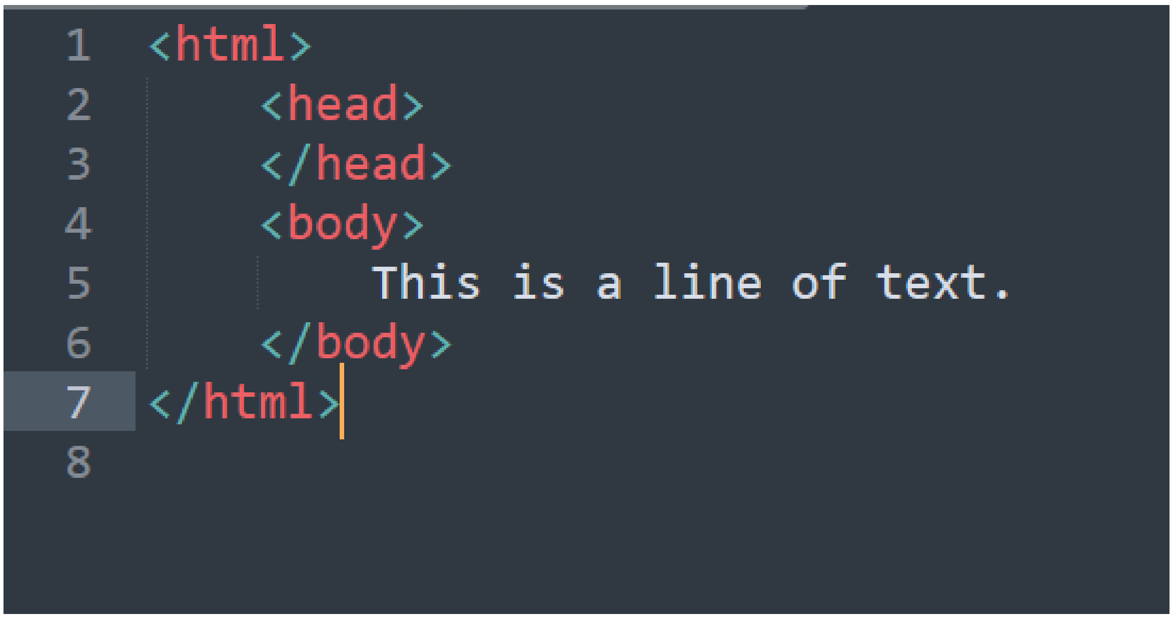
- Notepad
- Notepad++
- Sublime Text
- Atom Editor
- VS Code
- Froala
- Brackets
- Bluefish
- Komodo Edit
- Light Table
- Vim
- Netbeans

You can choose the one that works for you. For now let's write our examples in sublime text.



The HTML File

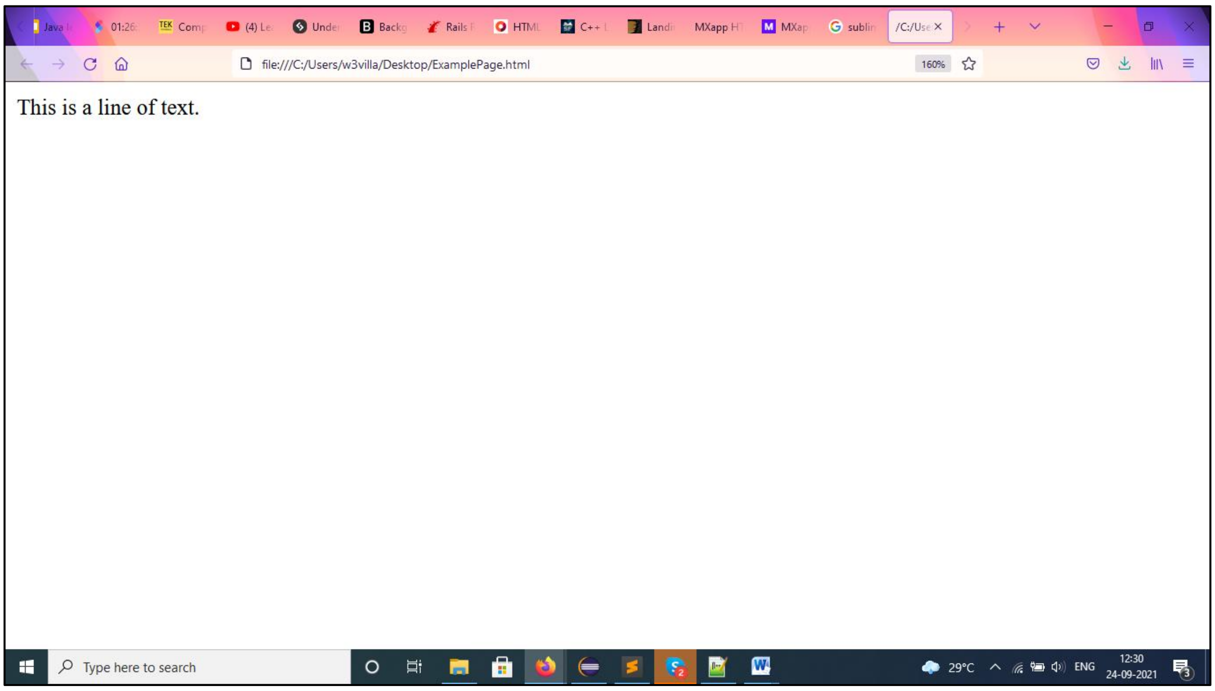
Add the basic HTML structure to the text editor with "This is a line of text" in the body section.



```
<html>
    <head>
    </head>
    <body>
        This is a line of text.
    </body>
</html>
```

In our example, the file is saved as first.html

When the file is opened, the following result is displayed in the web browser:

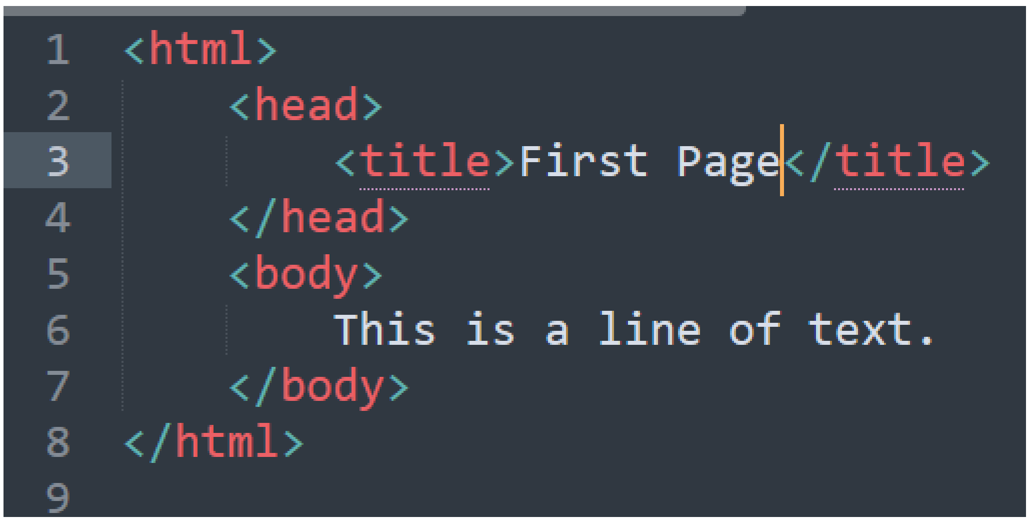

file:///C:/Users/w3villa/Desktop/ExamplePage.html

This is a line of text.

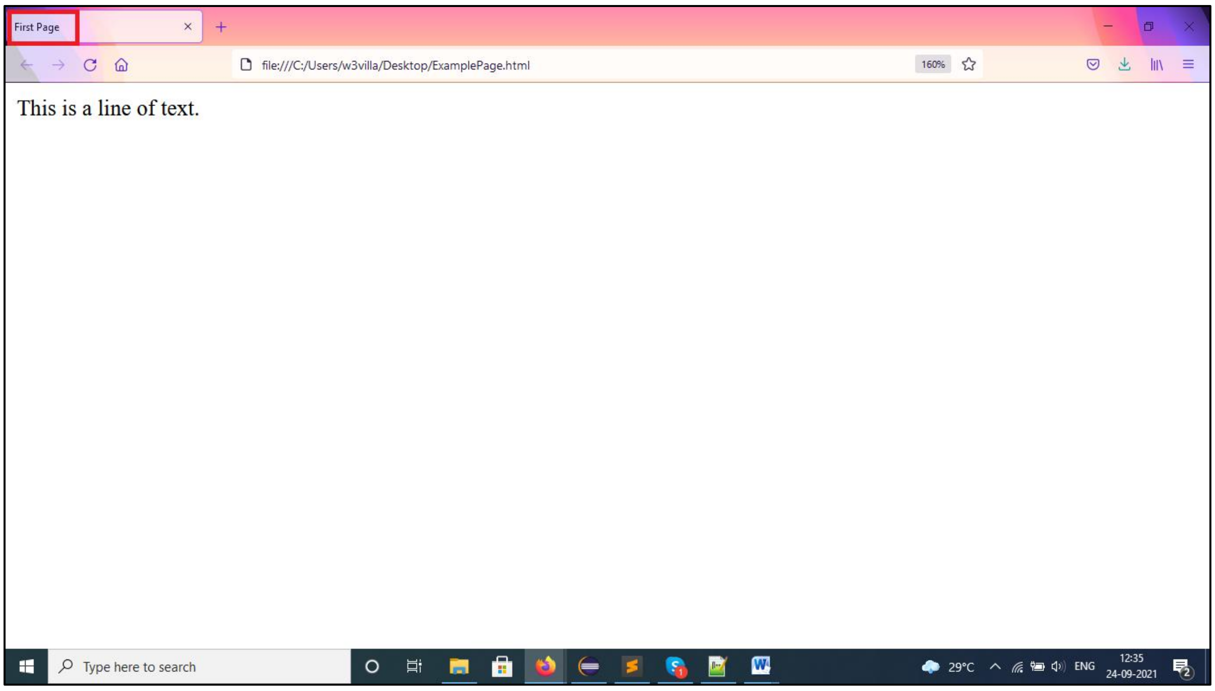
Don't forget to save the file. HTML file names should end in either **.html** or **.htm**

The <title> Tag

To place a title on the tab describing the web page, add a <title> element to your head section:


```
<html>
    <head>
        <title>First Page</title>
    </head>
    <body>
        This is a line of text.
    </body>
</html>
```

This will produce the following result:

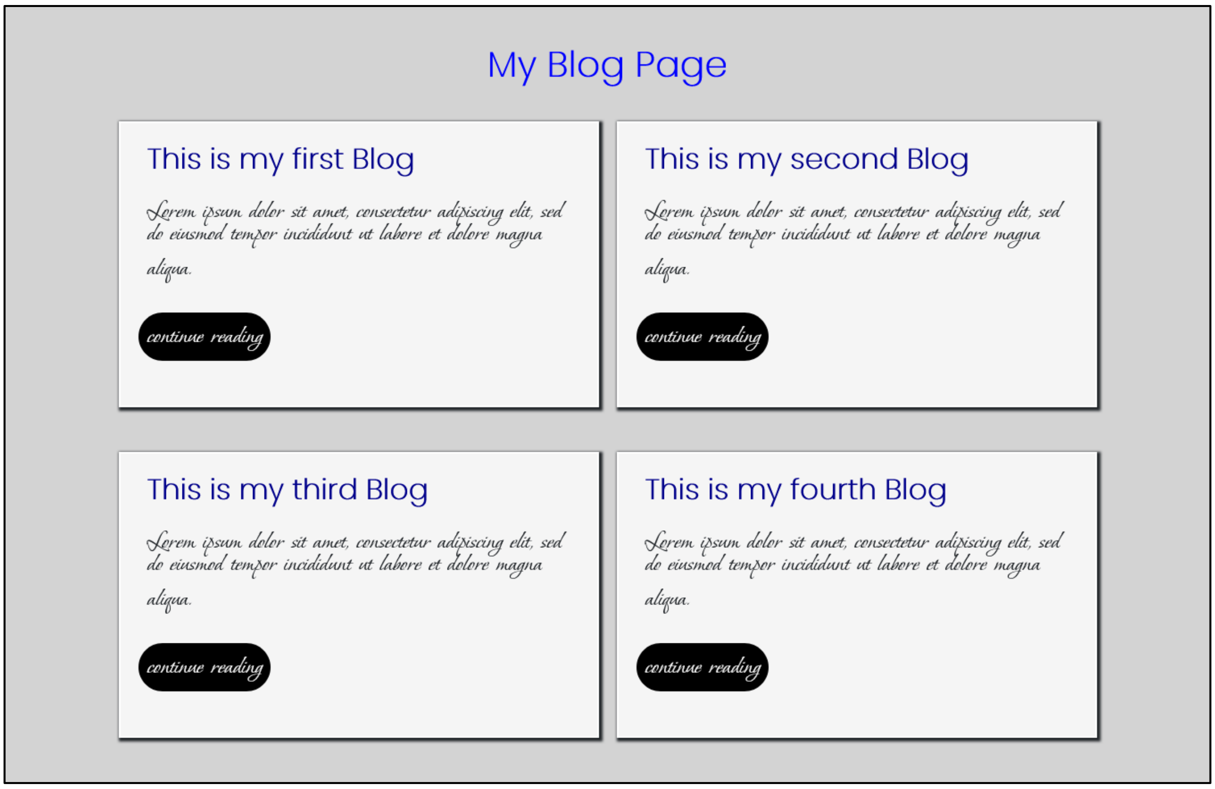


This is a line of text.

The title element is important because it describes the page and is used by search engines.

Creating a Blog

Throughout this course, we'll help you practice and create your own unique blog project, so you'll retain what you've learned and be able to put it to use. Just keep going and follow the instructions in the TASK section. This is what your finished blog page will look like.



My Blog Page

This is my first Blog

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

continue reading

This is my second Blog

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

continue reading

This is my third Blog

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

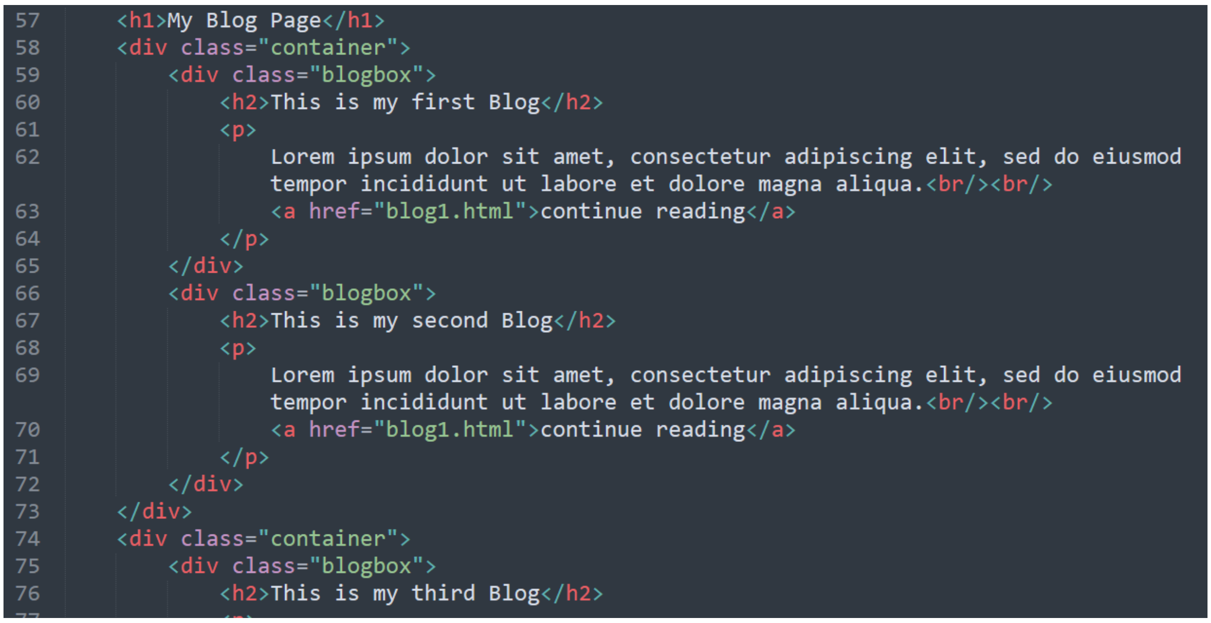
continue reading

This is my fourth Blog

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

continue reading

For the above blog page, the code looks



```
<h1>My Blog Page</h1>
<div class="container">
    <div class="blogbox">
        <h2>This is my first Blog</h2>
        <p>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
            tempor incididunt ut labore et dolore magna aliqua.<br/><br/>
            <a href="blog1.html">continue reading</a>
        </p>
    </div>
    <div class="blogbox">
        <h2>This is my second Blog</h2>
        <p>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
            tempor incididunt ut labore et dolore magna aliqua.<br/><br/>
            <a href="blog1.html">continue reading</a>
        </p>
    </div>
</div>
<div class="container">
    <div class="blogbox">
        <h2>This is my third Blog</h2>
```

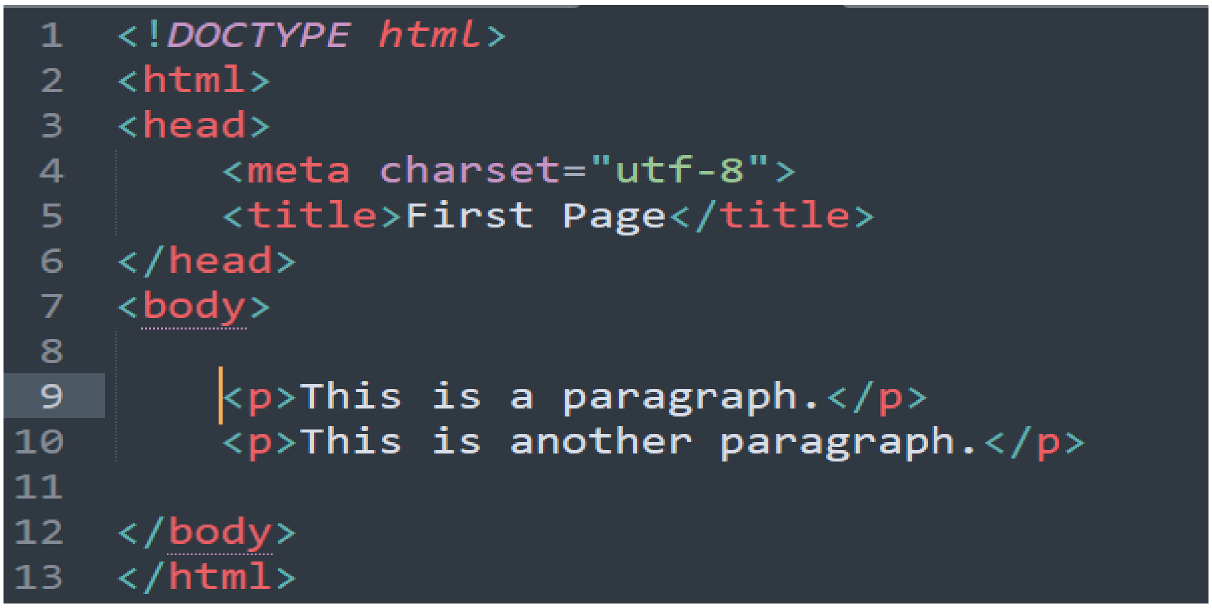
Don't be afraid of long codes. By the time you complete the course, everything will make complete sense and look highly doable.

TASK:

1. Open the text editor; create a new file named blog.html.
2. Create a heading named "My Blog Page"
3. Give the title to the page as "My Blog Page". Remember, the page title is located inside the <title> tag in the <head> of the page.

The <p> Element

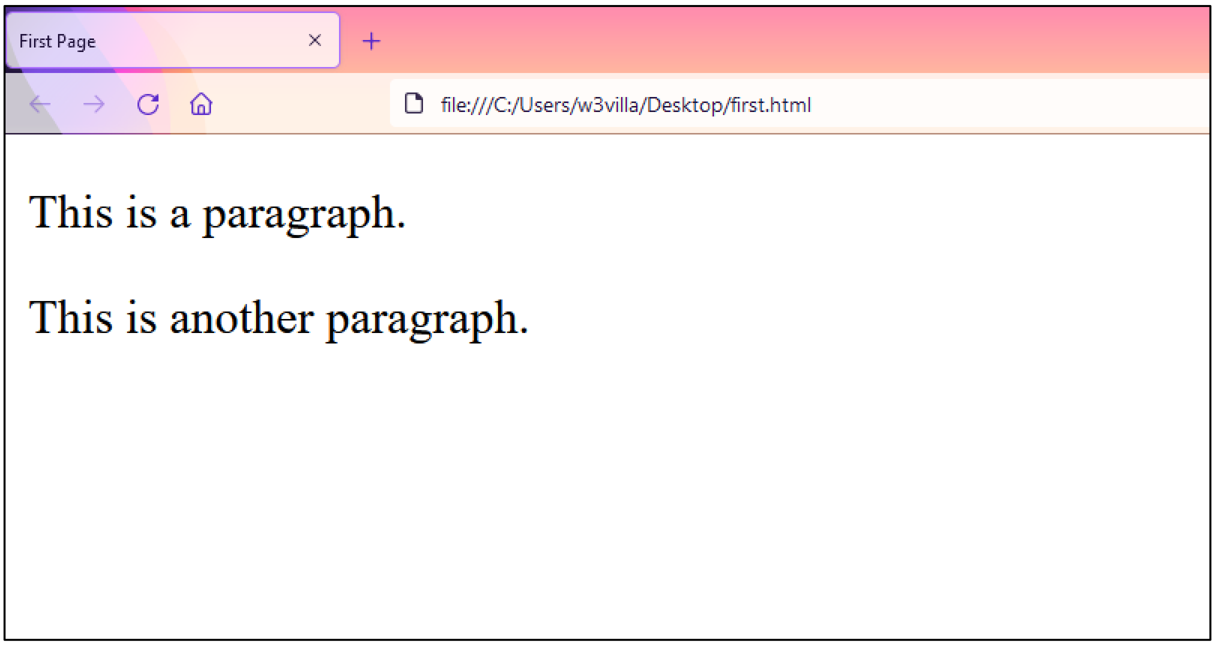
To create a paragraph, simply type in the <p> element with its opening and closing tags:



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>First Page</title>
</head>
<body>

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

</body>
</html>
```



This is a paragraph.

This is another paragraph.

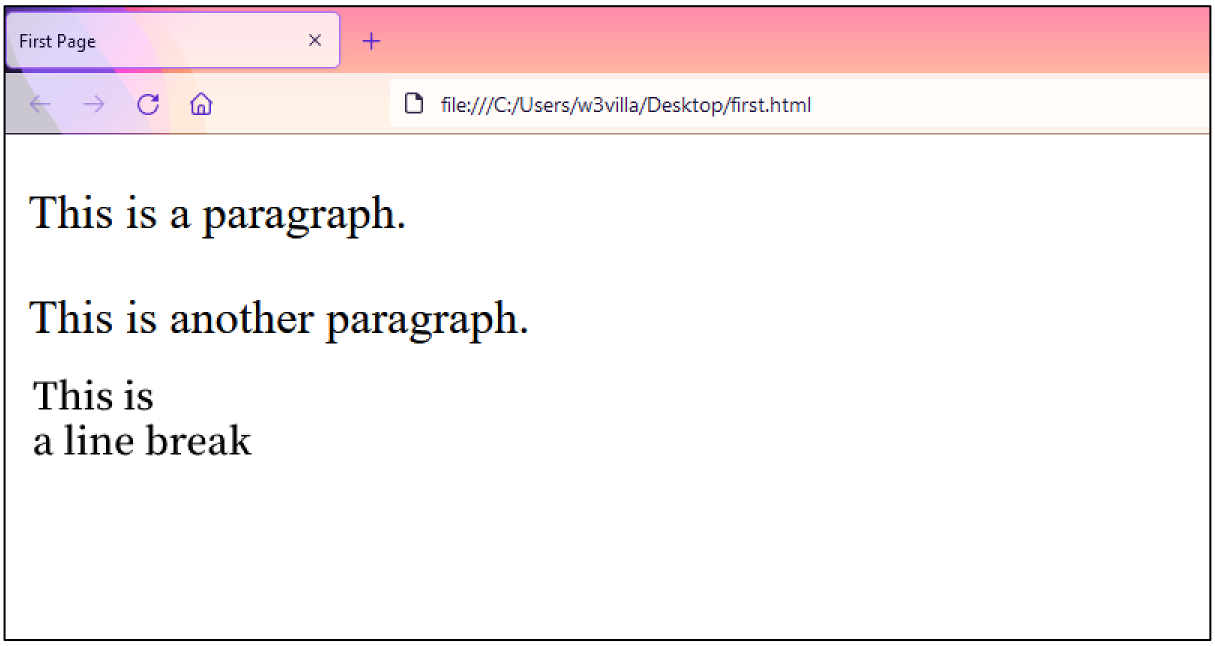
Browsers automatically add an empty line before and after a paragraph.

Single Line Break

Use the <br /> tag to add a single line of text without starting a new paragraph:

```
<html>
<head>
<title>first page</title>
</head>
<body>
<p>This is a paragraph.</p>
<p>This is another paragraph. </p>
<p>This is <br /> a line break </p>
</body>
</html>
```

Opening the HTML file in the browser shows that a single line break has been added to the paragraph:



This is a paragraph.

This is another paragraph.

This is
a line break

The <br /> element is an empty HTML element. It has no end tag.

Formatting Elements

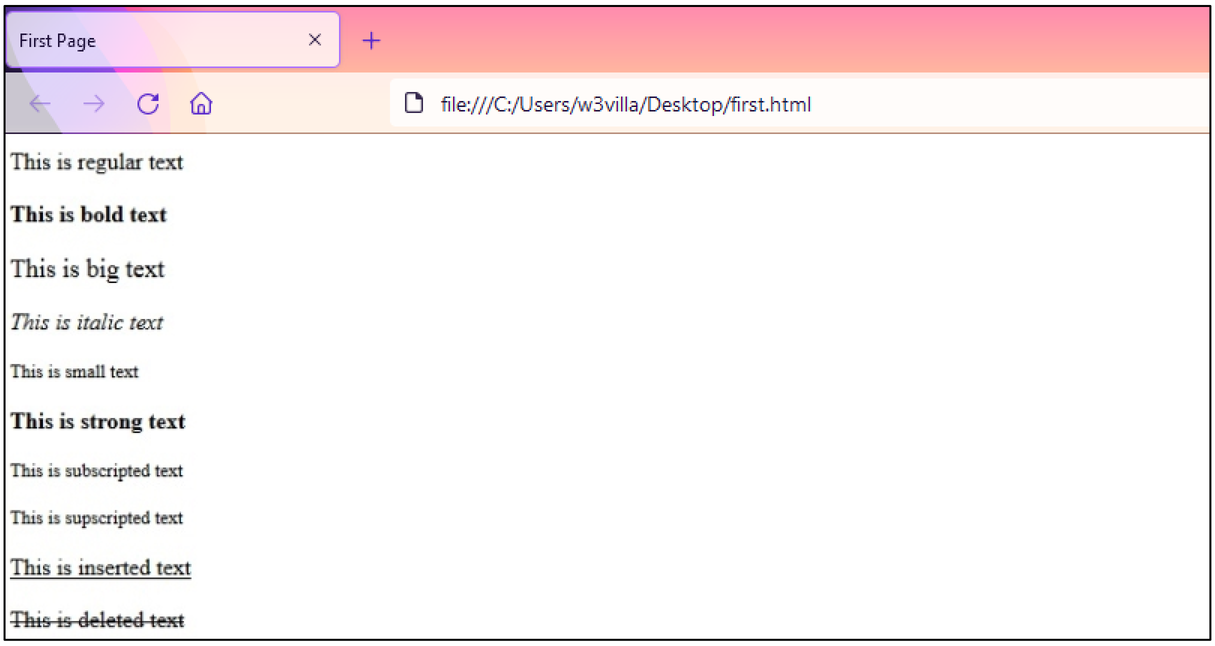
In HTML, there is a list of elements that specify text style.

Formatting elements were designed to display special types of text:

```
<html>
<head>
<title>first page</title>
</head>
<body>
<p>This is regular text </p>
<p><b> bold text </b></p>
<p><big> big text </big></p>
<p><i> italic text </i></p>
<p><small> small text </small></p>
<p><strong> strong text </strong></p>
<p><sub> subscripted text </sub></p>
<p><sup> superscripted text </sup></p>
<p><ins> inserted text </ins></p>
<p><del> deleted text </del></p>
</body>
</html>
```

The <strong> tag is a phrase tag. It defines important text.

Each paragraph in the example is formatted differently to demonstrate what each tag does:



This is regular text

This is bold text

This is big text

This is italic text

This is small text

This is strong text

This is subscripted text

This is supscripted text

This is inserted text

~~This is deleted text~~

Browsers display <strong> as <b>, and <em> as <i>.

However, the meanings of these tags differ: <b> and <i> define bold and italic text, respectively, while <strong> and <em> indicate that the text is "important".

HTML Headings

HTML includes six levels of headings, which are ranked according to importance.

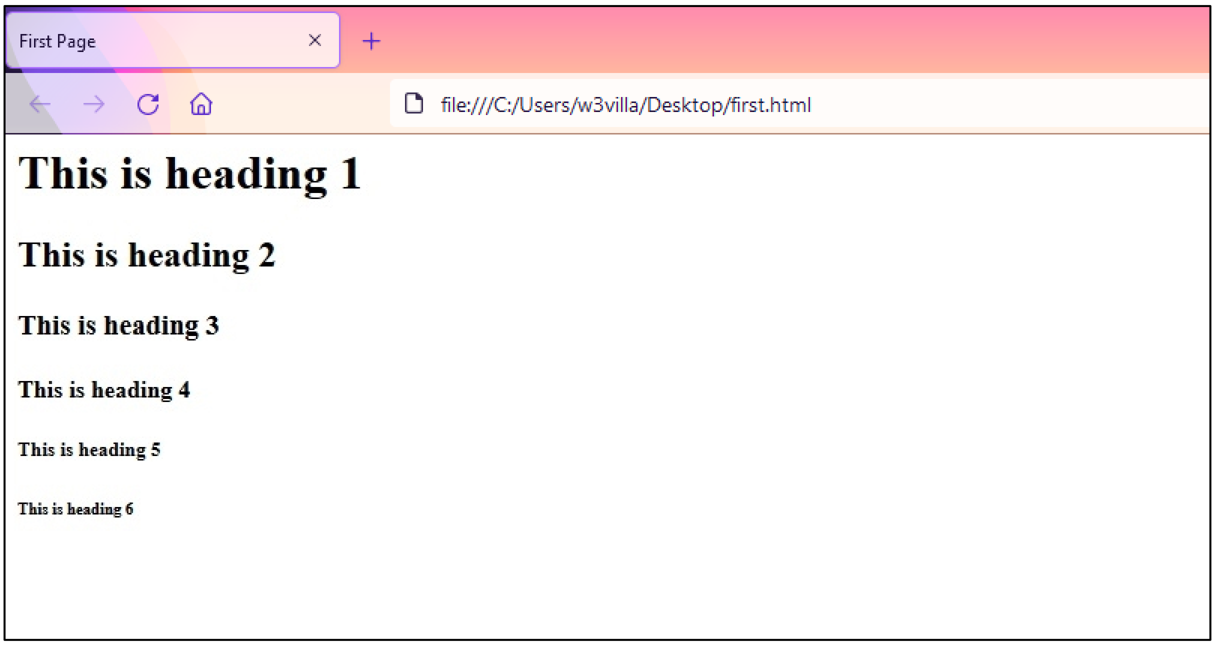
These are `<h1>, <h2>, <h3>, <h4>, <h5>, and <h6>`.

The following code defines all of the headings:

```
<html>
<head>
<title>first page</title>
</head>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
```

```
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

Result:



This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

It is not recommended that you use headings just to make the text big or bold, because search engines use headings to index the web page structure and content.

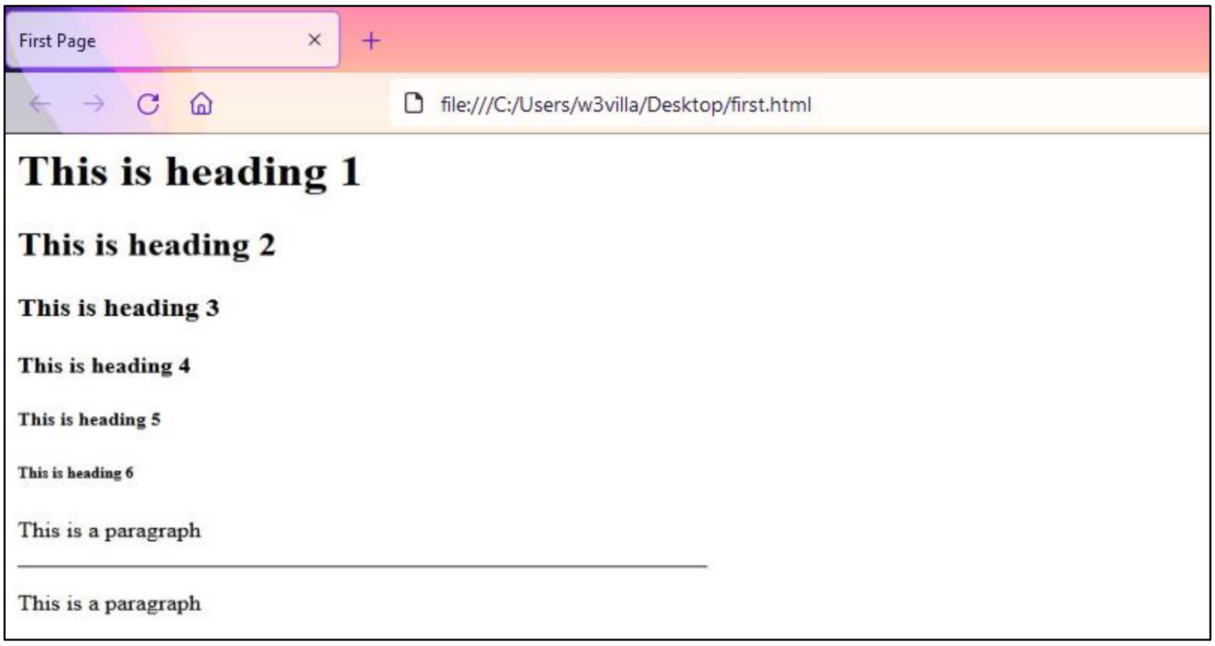
Horizontal Lines

To create a horizontal line, use the <hr /> tag.

```
<html>
<head>
<title>first page</title>
</head>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
<p>This is a paragraph </p>
```

```
<hr />
<p>This is a paragraph </p>
</body>
</html>
```

Result:



This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

This is a paragraph

This is a paragraph

In HTML5, the <hr> tag defines a thematic break.

Comments

The browser does not display comments, but they help document the HTML and add descriptions, reminders, and other notes.

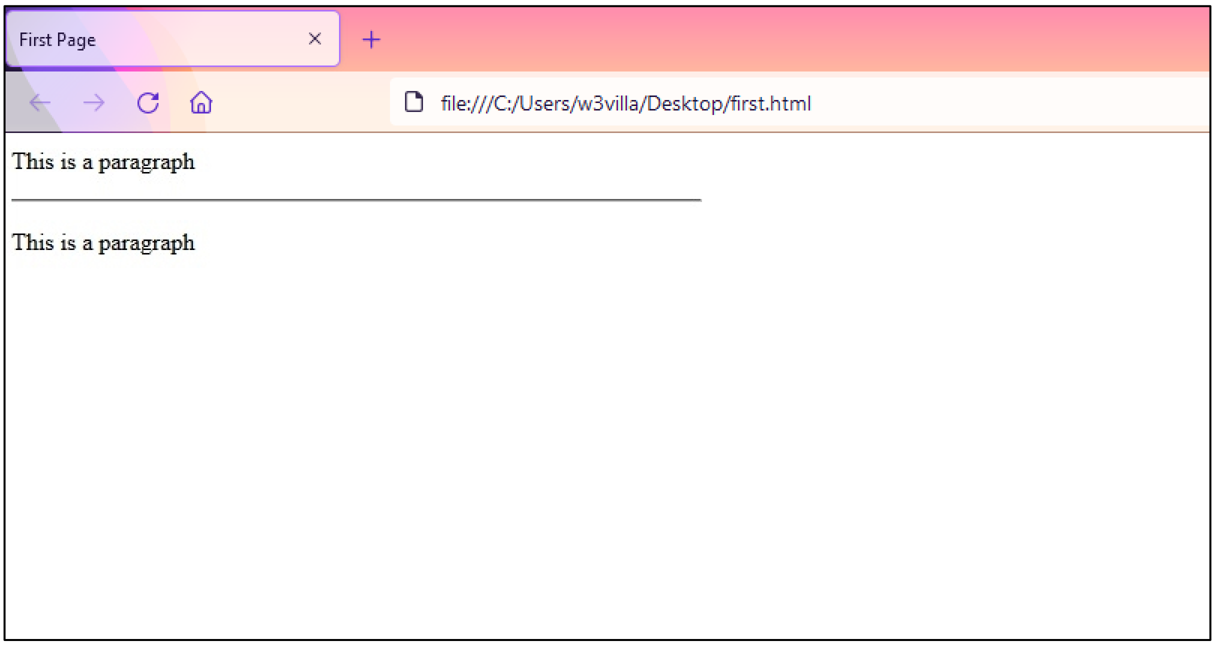
```
<!-- Your comment goes here -->
```

Example:

```
<html>
<head>
<title>first page</title>
</head>
<body>
<p>This is a paragraph </p>
<hr />
<p>This is a paragraph </p>
<!-- This is a comment -->
```

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

Result:


First Page

file:///C:/Users/w3villa/Desktop/first.html

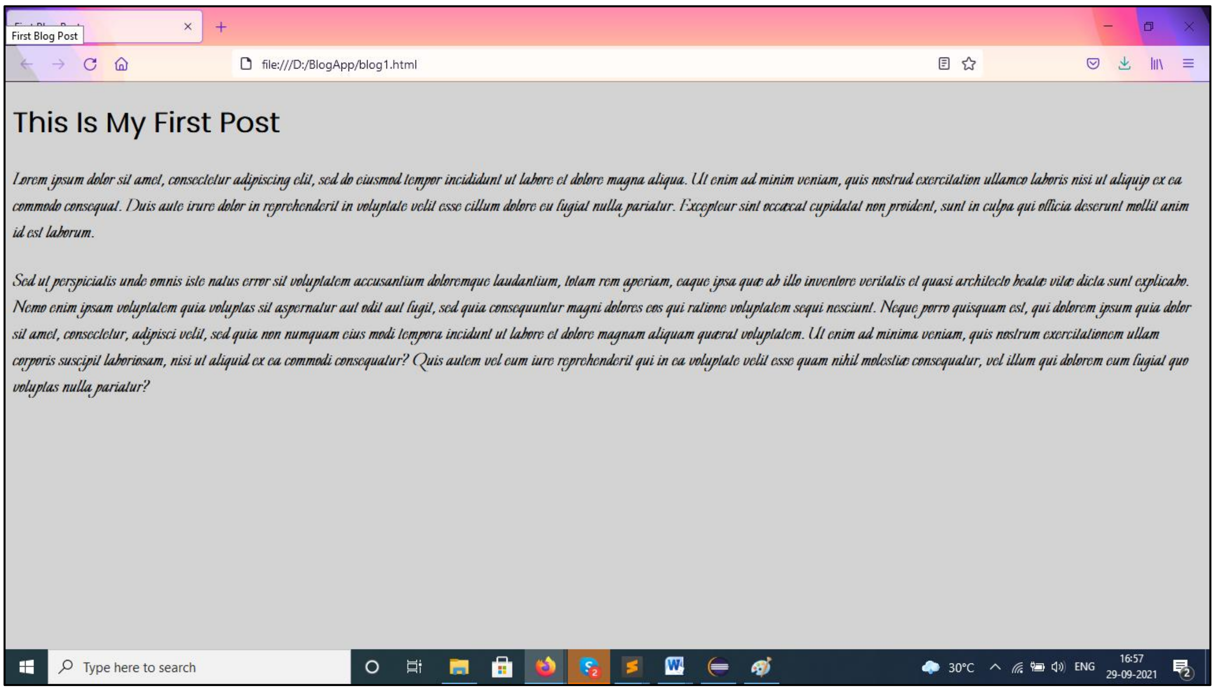
This is a paragraph

This is a paragraph

Formatting Text

Let's get back to our blog project.

Here, when we click on continue reading, we get a new page with title "First Blog Post". The page looks like this.


First Blog Post

file:///D:/BlogApp/blog1.html

This Is My First Post

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

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quæ ab illo inventore veritatis et quasi architecto beatæ vitæ dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quærat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiæ consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Let's take a look at the code:

```
<h1This is my first blog</h1>
<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua...
</p>
<p>
 Sed ut perspiciatis unde omnis iste natus error sit
voluptatem accusantium doloremque laudantium…..
</p>
```

You may have noticed that we've used some different fonts, as well as the colors in the page. Well this is done through CSS, which we will cover in our next tutorial!

TASK:

1. Create your own blog page, create a heading using <h1> tag, and paragraphs using <p> tags. Apply the formatting that you have learned so far.

2. Play around with the code; experiment with more formatting, and applying colors, chaging fonts etc.

HTML Elements

HTML documents are made up of HTML elements.

An HTML element is written using a start tag and an end tag, and with the content in between.

HTML documents consist of nested HTML elements. In the example below, the body element includes the <p> tags, the <br /> tag and the content, "This is a paragraph".

```
<html>
<head>
<title>first page</title>
</head>
```

```
<body>
<p>This is a paragraph <br /></p>
</body>
</html>
```

Some HTML elements (like the <br /> tag) do not have end tags.
Some elements are quite small. Since you can't put contents within a break tag, and you don't have an opening and closing break tag, it's a separate, single element.

So HTML is really scripting with elements within elements.

```
<html>
<head>
<title>first page</title>
</head>
<body>
<p>This is a paragraph</p>
<p>This is a <br /> line break</p>
</body>
</html>
```

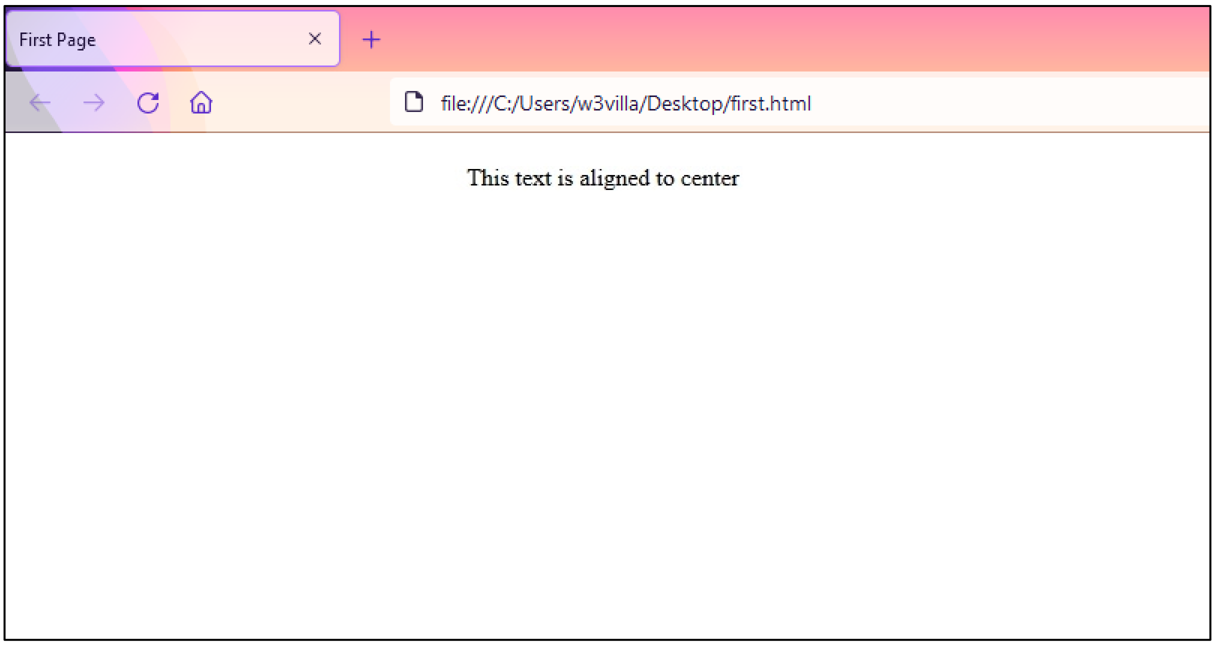
Some HTML elements (like the <br /> tag) do not have end tags.

HTML Attributes

Attributes provide additional information about an element or a tag, while also modifying them. Most attributes have a value; the value modifies the attribute.

```
<p align="center">
This text is aligned to center
</p>
```

In this example, the value of "center" indicates that the content within the p element should be aligned to the center:



This text is aligned to center

Attributes are always specified in the start tag, and they appear in name="value" pairs.

Attribute Measurements

As an example, we can modify the horizontal line so it has a width of 50 pixels.

This can be done by using the width attribute:

```
<hr width="50px" />
```

An element's width can also be defined using percentages:

```
<hr width="50%" />
```

An element's width can be defined using pixels or percentages.

The Align Attribute

The align attribute is used to specify how the text is aligned.

In the example below, we have a paragraph that is aligned to the center, and a line that is aligned to the right.

```
<html>
<head>
```

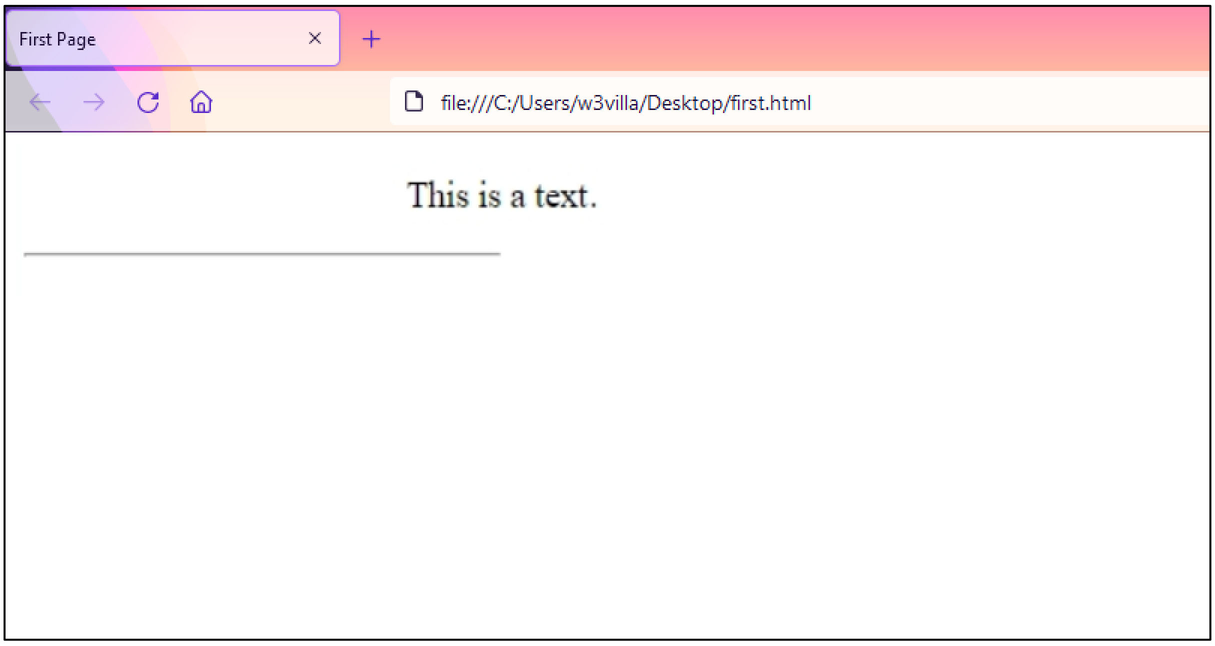
```
<title>Attributes</title>
</head>
<body>
<p align="center">This is a text <br />
<hr width="10%" align="right" /> This is also a text.
</p>
</body>
</html>
```

The align attribute of <p> is not supported in HTML5.

Attributes

You may be wondering what happens if you try to apply contradictory attributes within the same element.

```
<p align="center">
This is a text.
<hr width="50%" align="left" />
</p>
```



This is a text.

The align attribute of <p> is not supported in HTML5.

The <img> Tag

The <img> tag is used to insert an image. It contains only attributes, and does not have a closing tag.

The image's URL (address) can be defined using the src attribute.

The HTML image syntax looks like this:

```
<img src="image.jpg" />
```

The alt attribute specifies an alternate text for an image.
Image Location

You need to put in the image location for the src attribute that is between the quotation marks.

For example, if you have a photo named "tree.jpg" in the same folder as the HTML file, your code should look like this:

```
<html>
<head>
<title>first page</title>
</head>
<body>
<img src="tree.jpg" alt="" />
</body>
</html>
```

In case the image cannot be displayed, the alt attribute specifies an alternate text that describes the image in words. The alt attribute is required.
Image Resizing

To define the image size, use the width and height attributes.
The value can be specified in pixels or as a percentage:

```
<html>
<head>
<title>first page</title>
```

```
</head>
<body>
<img src="tree.jpg" height="150px" width="150px" alt="" />
<!-- or -->
<img src="tree.jpg" height="50%" width="50%" alt="" />
</body>
</html>
```

Loading images takes time. Using large images can slow down your page, so use them with care.

Image Border

By default, an image has no borders. Use the border attribute within the image tag to create a border around the image.

```
<img src="tree.jpg" height="150px" width="150px"
border="1px" alt="" />
```

By default, Internet Explorer 9, and its earlier versions, display a border around an image unless a border attribute is defined.

The <a> Tag

Links are also an integral part of every web page. You can add links to text or images that will enable the user to click on them in order to be directed to another file or webpage.

In HTML, links are defined using the <a> tag.

Use the href attribute to define the link's destination address:

```
<a href=""></a>
```

To link an image to another document, simply nest the <img> tag inside <a> tags.

Creating Your First Link

In the example below, a link to W3grads website is defined:

```
<a href="https://www.w3grads.com"> Learn Coding </a>
```

Once the code has been saved, "Learn Coding" will display as a link:

Learn Coding

Clicking on "Learn Coding" redirects you to www.w3grads.com

Links can be either absolute or relative.

The target Attribute

The target attribute specifies where to open the linked document.
Giving a _blank value to your attribute will have the link open in a new window or new tab:

```
<a href="https://www.w3grads.com" target="_blank">
Learn Coding
</a>
```

A visited link is underlined and purple.

HTML Ordered Lists

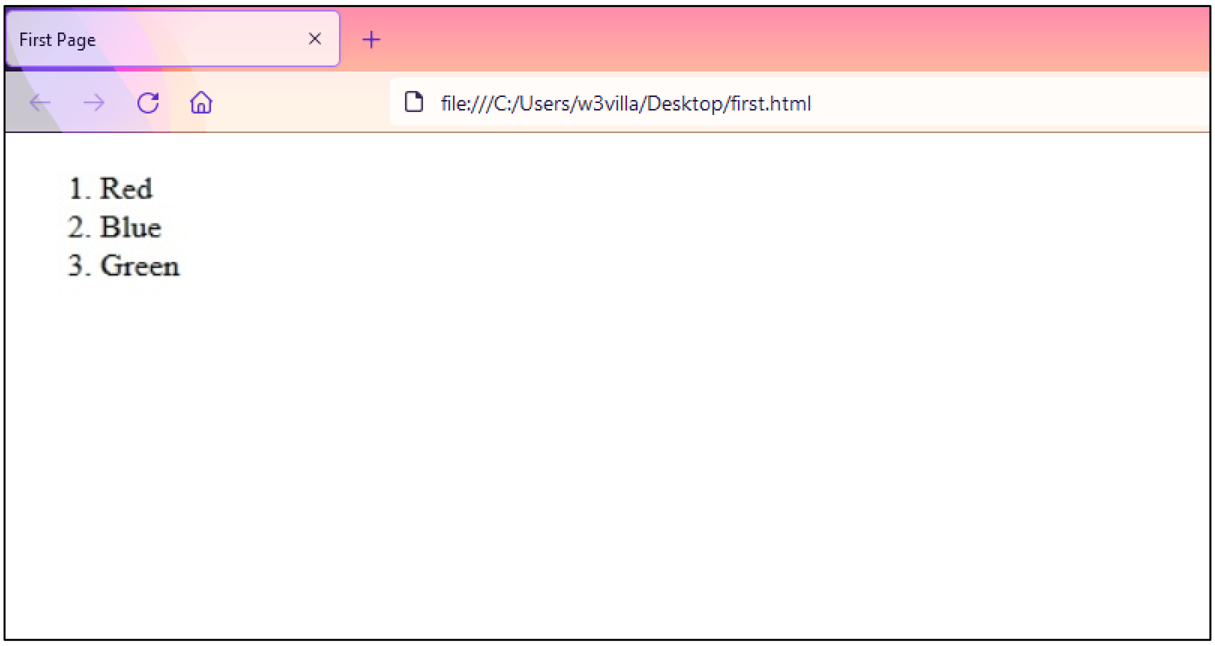
An ordered list starts with the <ol> tag, and each list item is defined by the <li> tag.

Here is an example of an ordered list:

```
<html>
<head>
<title>first page</title>
</head>
<body>
```

```
<ol>
<li>Red</li>
<li>Blue</li>
<li>Green</li>
</ol>
</body>
</html>
```

Result:



First Page

file:///C:/Users/w3villa/Desktop/first.html

1. Red
2. Blue
3. Green

The list items will be automatically marked with numbers.

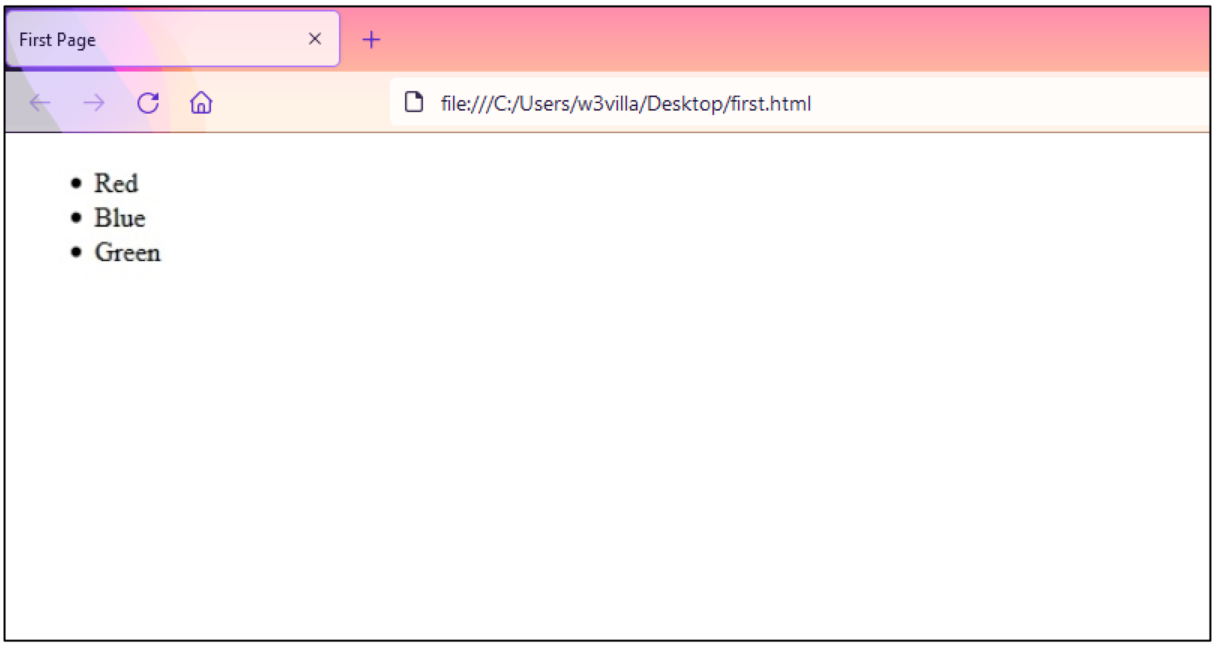
HTML Unordered List

An unordered list starts with the <ul> tag.

```
<html>
<head>
<title>first page</title>
</head>
<body>
<ul>
<li>Red</li>
<li>Blue</li>
<li>Green</li>
</ul>
</body>
```

```
</html>
```

Result:



- Red
- Blue
- Green

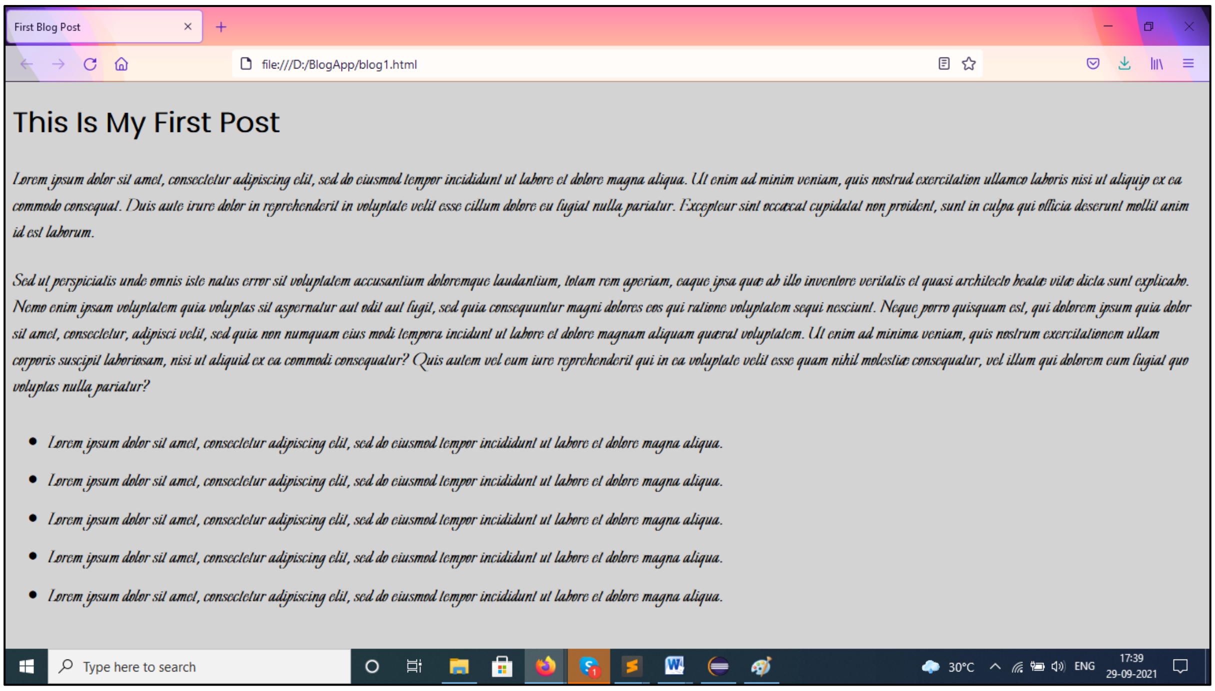
The list items will be marked with bullets.

Add pointers in the blog

Back to our blog! Let's create some pointers in our existing blog page.

Reminder: Use the <ul> tag, in which each item is represented by the <li> tag, to create an unordered list. Take a look at the code:

```
<ul>
<li>Lorem ipsum dolor sit amet, consectetur adipiscing
elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua.</li>
<li>Lorem ipsum dolor sit amet, consectetur adipiscing
elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua.</li>
<li>Lorem ipsum dolor sit amet, consectetur adipiscing
elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua.</li>
…
…
</ul>
```



This Is My First Post

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

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

- *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*
- *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*
- *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*
- *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*
- *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*

TASK:

1. Add your own bullet points in your article.

2. Use the <a> tag in the list items to create hyperlinks

Creating a Table

Tables are defined by using the <table> tag.

Tables are divided into table rows with the <tr> tag.

Table rows are divided into table columns (table data) with the <td> tag.

Here is an example of a table with one row and three columns:

```
<table>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</table>
```

Table data tags <td> act as data containers within the table.

They can contain all sorts of HTML elements, such as text, images, lists, other tables, and so on.

The *border* and *colspan* Attributes

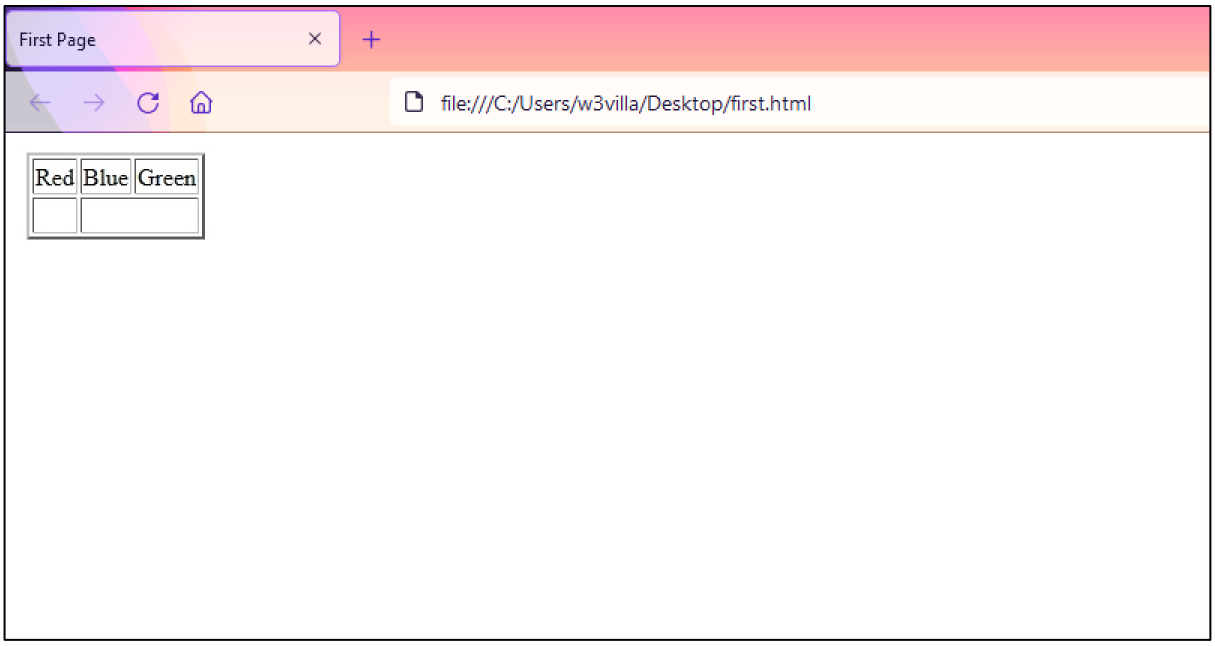
A border can be added using the border attribute:

```
<table border="2">
```

A table cell can span two or more columns:

```
<table border="2">
<tr>
<td>Red</td>
<td>Blue</td>
<td>Green</td>
</tr>
<tr>
<td><br /></td>
<td colspan="2"><br /></td>
</tr>
</table>
```

Result:



Red	Blue	Green

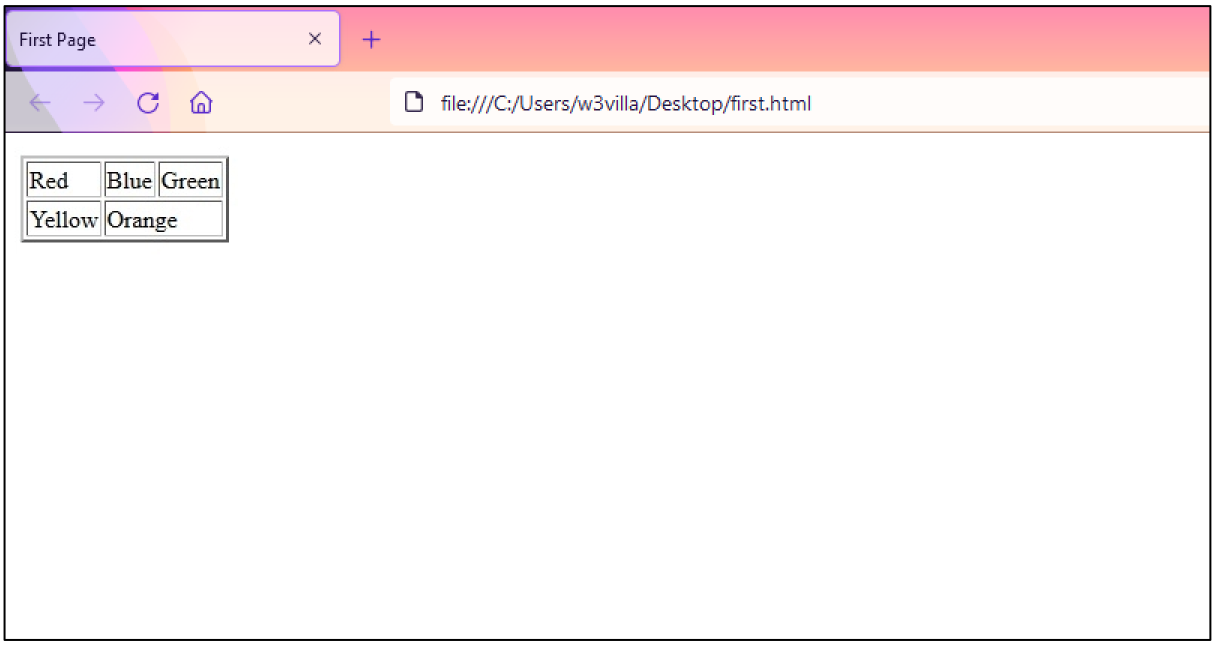
The border attribute is not supported in HTML5.

Colspan Color

The example below demonstrates the colspan attribute in action:

```
<table border="2">
<tr>
<td>Red</td>
<td>Blue</td>
<td>Green</td>
</tr>
<tr>
<td>Yellow</td>
<td colspan="2">Orange</td>
</tr>
</table>
```

Result:



Red	Blue	Green
Yellow	Orange	

You can see that the cell containing "Orange" spans two cells. To make a cell span more than one row, use the rowspan attribute.

The align and bgcolor Attributes

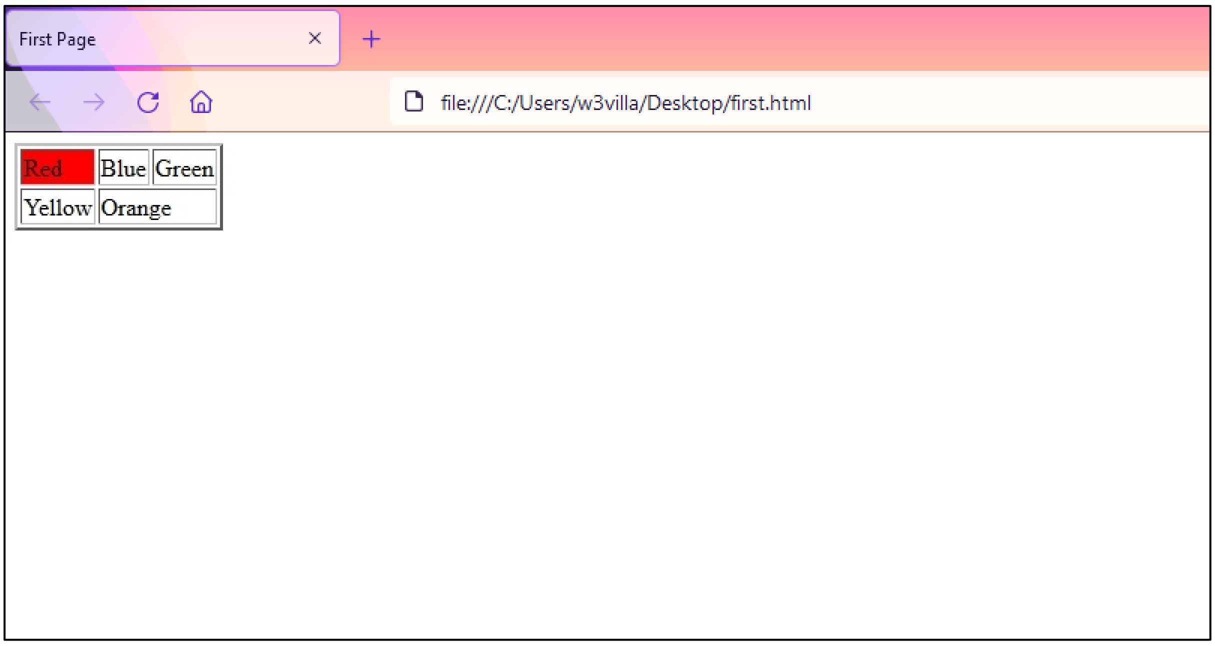
To change your table's position, use the align attribute inside your table tag:

```
<table align="center">
```

Now let's specify a background color of red for a table cell. To do that, just use the bgcolor attribute.

```
<table border="2">
<tr>
<td bgcolor="red">Red</td>
<td>Blue</td>
<td>Green</td>
</tr>
<tr>
<td>Yellow</td>
<td colspan="2">Orange</td>
</tr>
</table>
```

Result:



🟥 Red	Blue	Green
Yellow	Orange	

In the case of styling elements, CSS is more effective than HTML. Try our free "Learn CSS" course to learn more about CSS and styles.

Types of Elements

In HTML, most elements are defined as block level or inline elements.

Block level elements start from a new line. For example: `<h1>, <form>, <li>, <ol>, <ul>, <p>, <pre>, <table>, <div>`, etc.

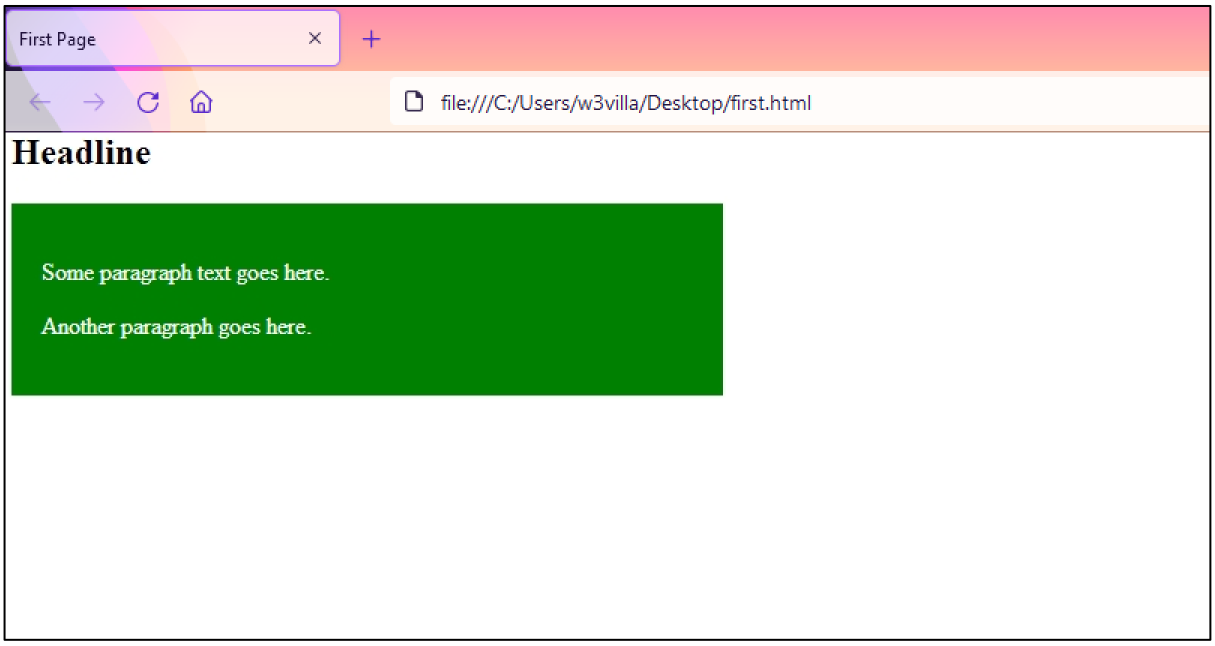
Inline elements are normally displayed without line breaks. For example: `<b>`, `<a>`, `<strong>`, `<img>`, `<input>`, `<em>`, `<span>`, etc.

The <div> element is a block-level element that is often used as a container for other HTML elements.

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

```
<html>
<body>
<h1>Headline</h1>
<div style="background-color:green; color:white;
padding:20px;">
<p>Some paragraph text goes here.</p>
<p>Another paragraph goes here.</p>
</div>
</body>
</html>
```

Result:



Headline

Some paragraph text goes here.

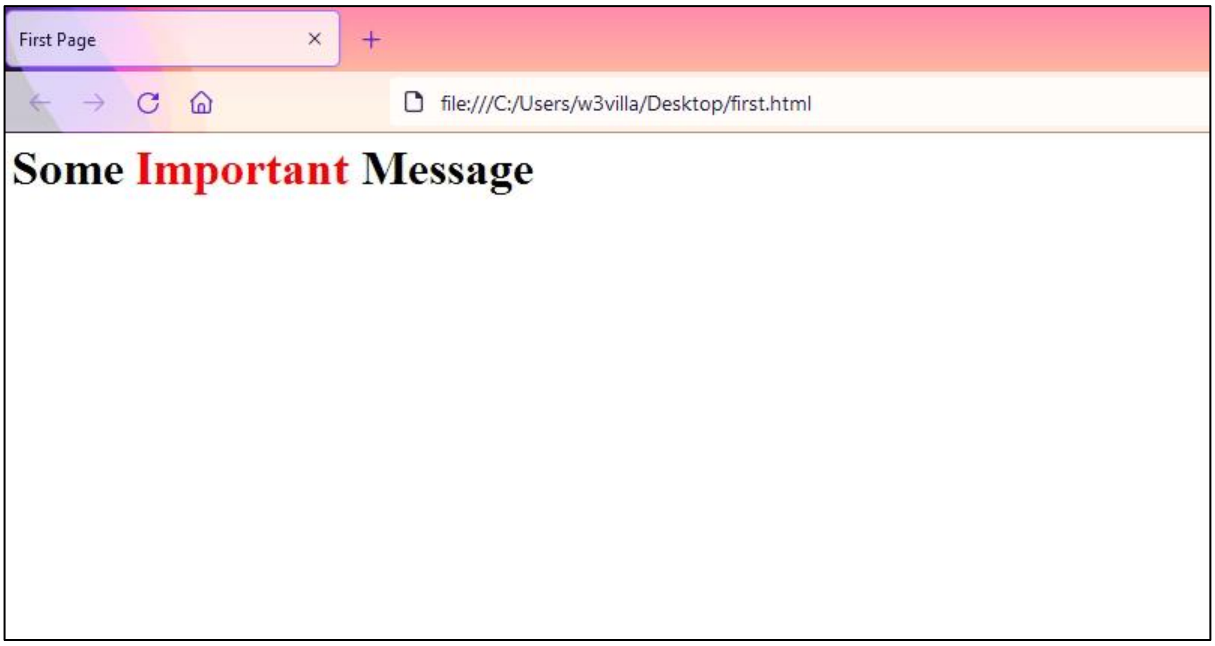
Another paragraph goes here.

Similarly, the <span> element is an inline element that is often used as a container for some text.

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

```
<html>
<body>
<h2>Some
<span style="color:red">Important</span>
Message</h2>
</body>
</html>
```

Result:



Some Important Message

Other elements can be used either as block level elements or inline elements.

This includes the following elements:

APPLET - embedded Java applet
IFRAME - Inline frame
INS - inserted text
MAP - image map
OBJECT - embedded object
SCRIPT - script within an HTML document

You can insert inline elements inside block elements. For example, you can have multiple <span> elements inside a <div> element.

Inline elements cannot contain any block level elements.

The <form> Element

HTML forms are used to collect information from the user.

Forms are defined using the <form> element, with its opening and closing tags:

```
<body>
<form>…</form>
</body>
```

Use the action attribute to point to a webpage that will load after the user submits the form.

```
<form action="http://www.w3grads.com">
</form>
```

Usually the form is submitted to a web page on a web server.

The method and name Attributes

The method attribute specifies the HTTP method (GET or POST) to be used when forms are submitted (see below for description):

```
<form action="url" method="GET">

<form action="url" method="POST">
```

When you use GET, the form data will be visible in the page address.

Use POST if the form is updating data, or includes sensitive information (passwords).

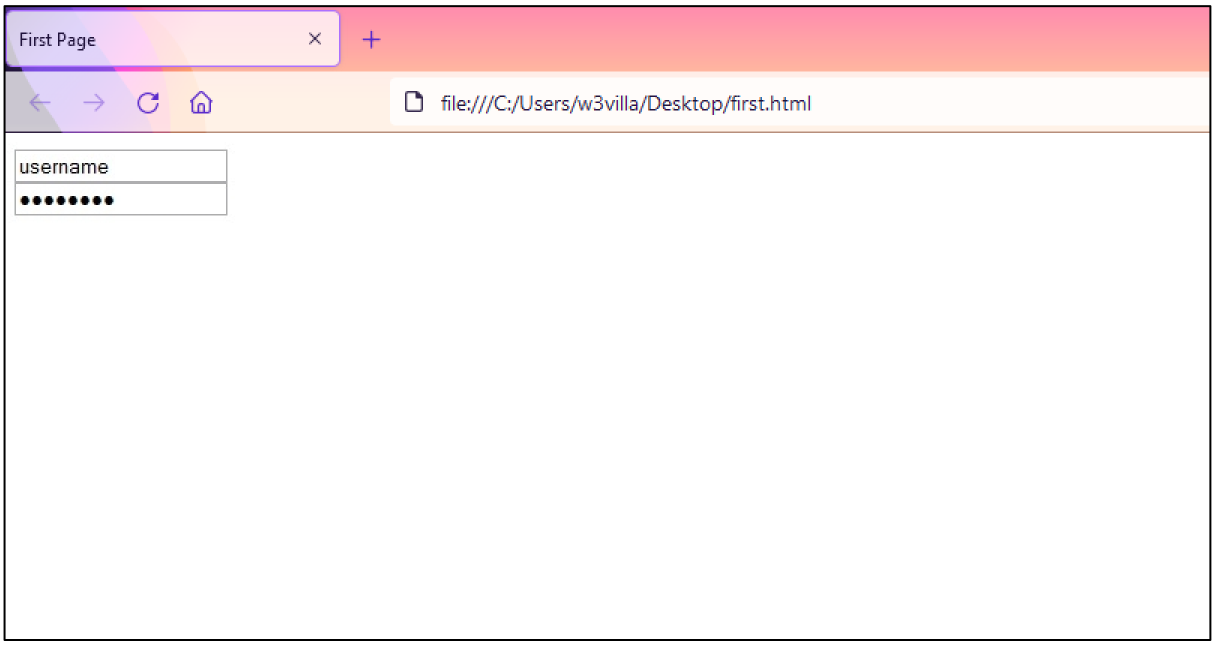
POST offers better security because the submitted data is not visible in the page address.

To take in user input, you need the corresponding form elements, such as text fields. The <input> element has many variations, depending on the type attribute. It can be a text, password, radio, URL, submit, etc.

The example below shows a form requesting a username and password:

```
<form>
<input type="text" name="username" /><br />
<input type="password" name="password" />
</form>
```

Result:



The name attribute specifies a name for a form.

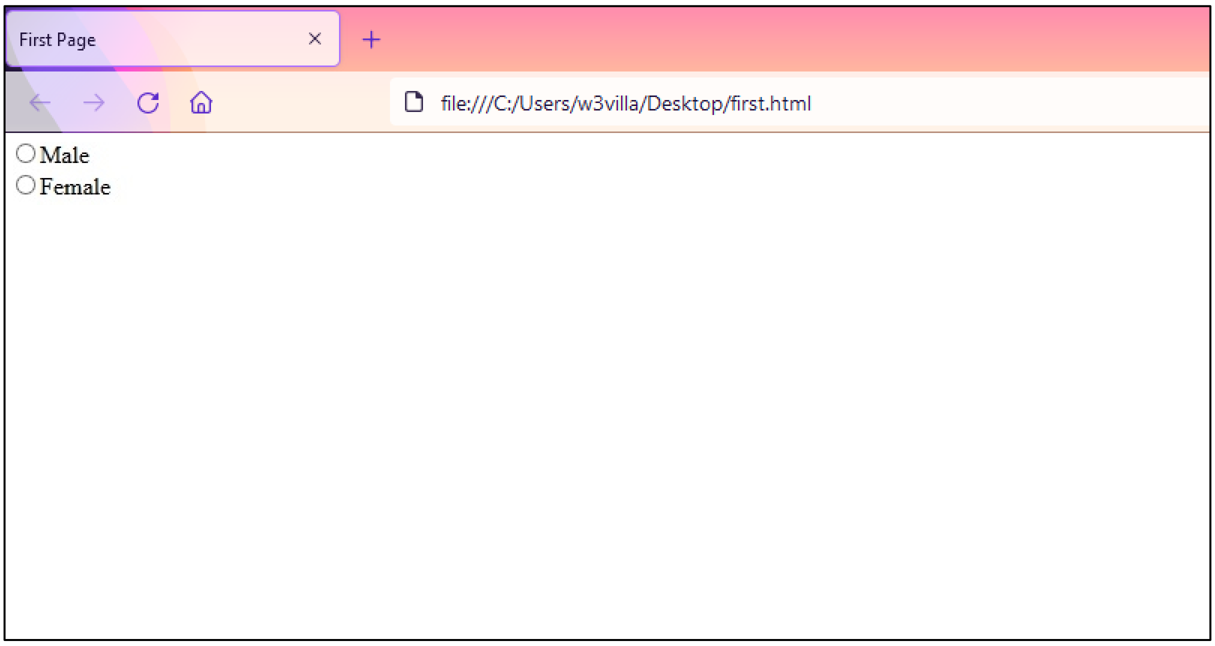
Form Elements

If we change the input type to radio, it allows the user select only one of a number of choices:

```
<input type="radio" name="gender" value="male" /> Male <br
/>
```

```
<input type="radio" name="gender" value="female" /> Female
<br />
```

Result:



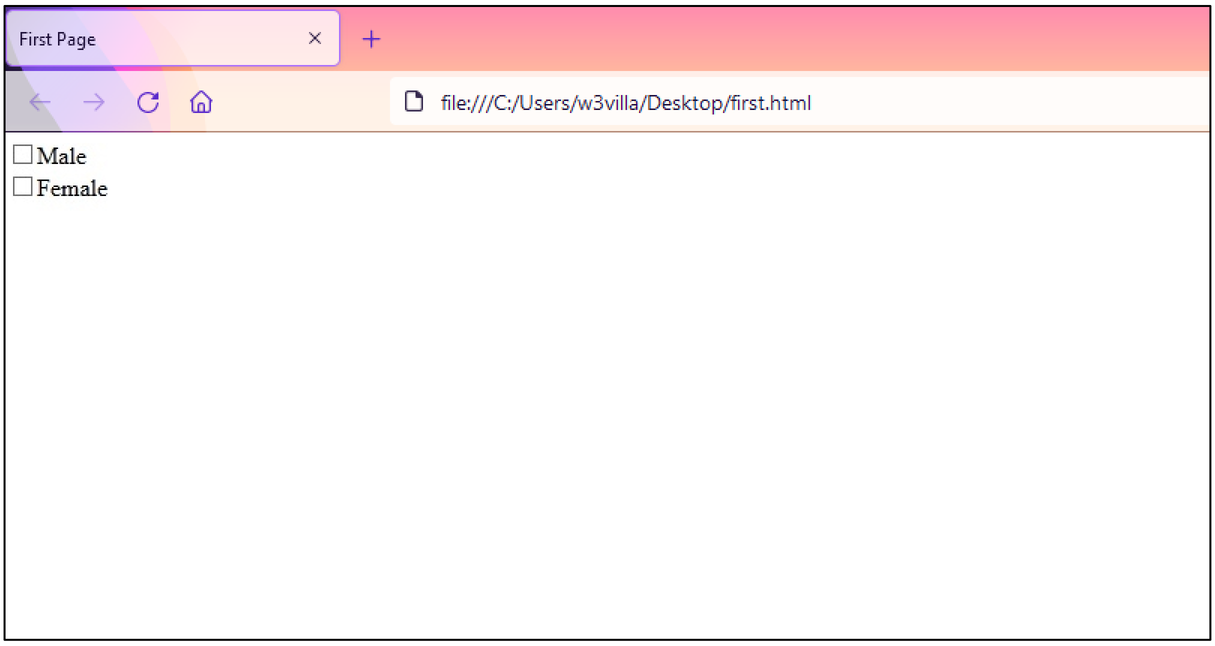
○Male
○Female

The type "checkbox" allows the user to select more than one option:

```
<input type="checkbox" name="gender" value="1" /> Male <br
/>

<input type="checkbox" name="gender" value="2" /> Female
<br />
```

Result:



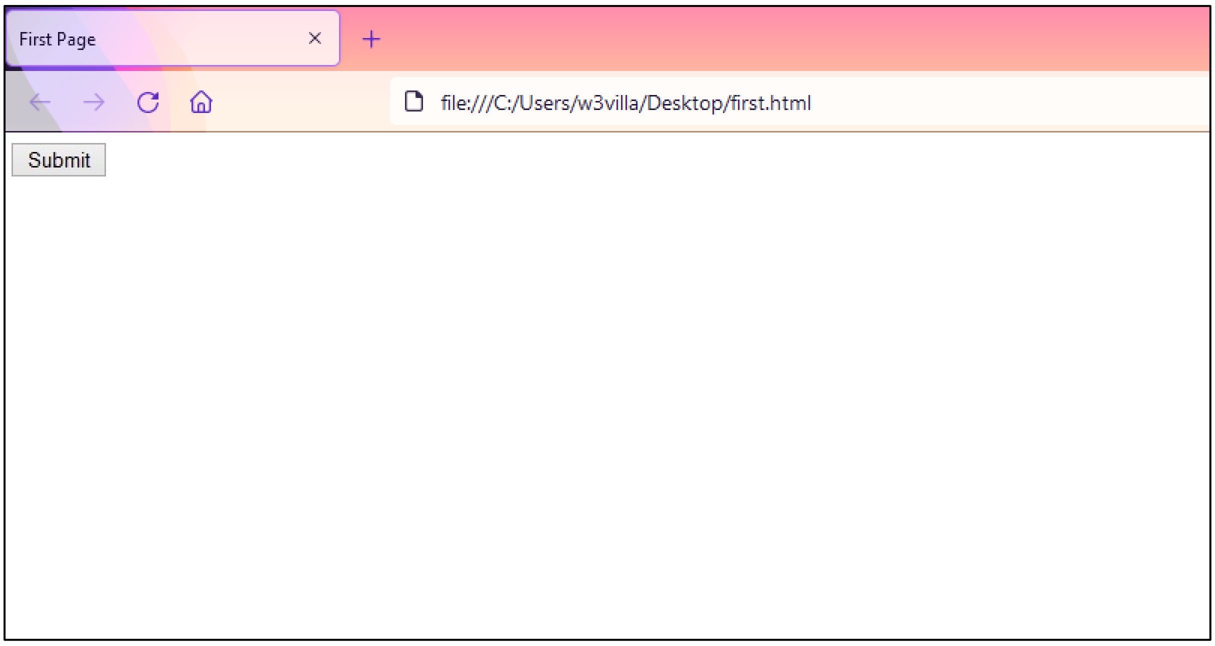
☐Male
☐Female

The <input> tag has no end tag.

The submit button submits a form to its action attribute:

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

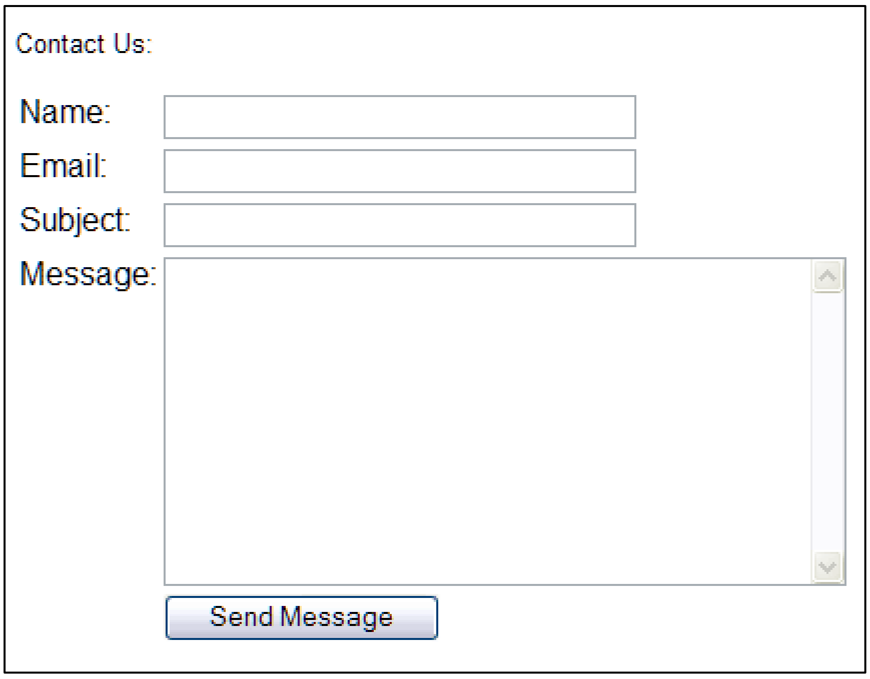
Result:



After the form is submitted, the data should be processed on the server using a programming language, such as PHP, Node.js, Ruby, Java, etc.

Assignment - Contact Form

This is an assignment task, where you will need to create a contact form that will include the information name, message fields etc. The completed form should look like this.



This is just a static HTML page, so it won't work to actually submit the form. You'd need to create the server-side code in order to submit a real form and process the data. You can learn any server side programming language in order to get this form working, once you've completed the HTML and CSS courses.

HTML Colors!

HTML colors are expressed as hexadecimal values.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

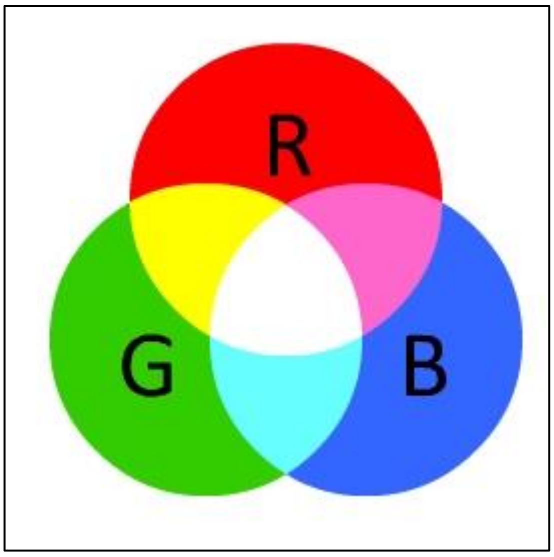
As you can see, there are 16 values there, 0 through F. Zero represents the lowest value, and F represents the highest.

HTML Color Model

Colors are displayed in combinations of red, green, and blue light (RGB).

Hex values are written using the hashtag symbol (#), followed by either three or six hex characters.

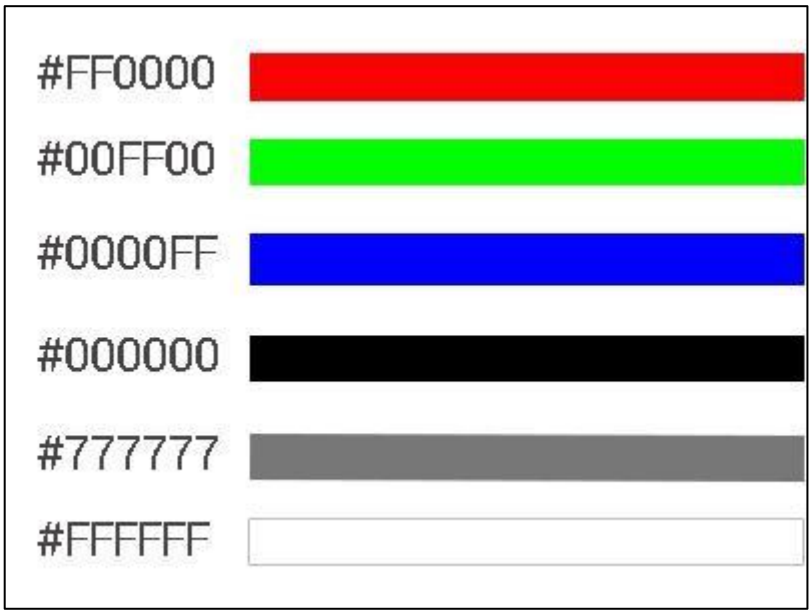
As shown in the picture below, the circles overlap, forming new colors:



RGB color values are supported in all browsers.

Color Values

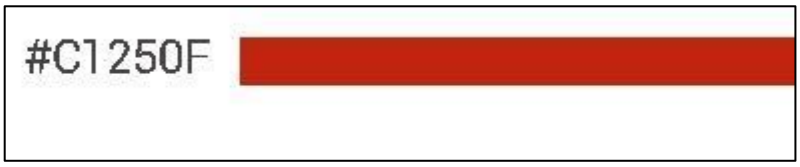
All of the possible red, green, and blue combinations potentially number over 16 million. Here are only a few of them:



#FF0000	🟥
#00FF00	🟩
#0000FF	🟦
#000000	⬛
#777777	▨
#FFFFFF	⬜

We can mix the colors to form additional colors:

Orange and red mix:



#C1250F	🟥

Hexadecimal color values are supported in all browsers.

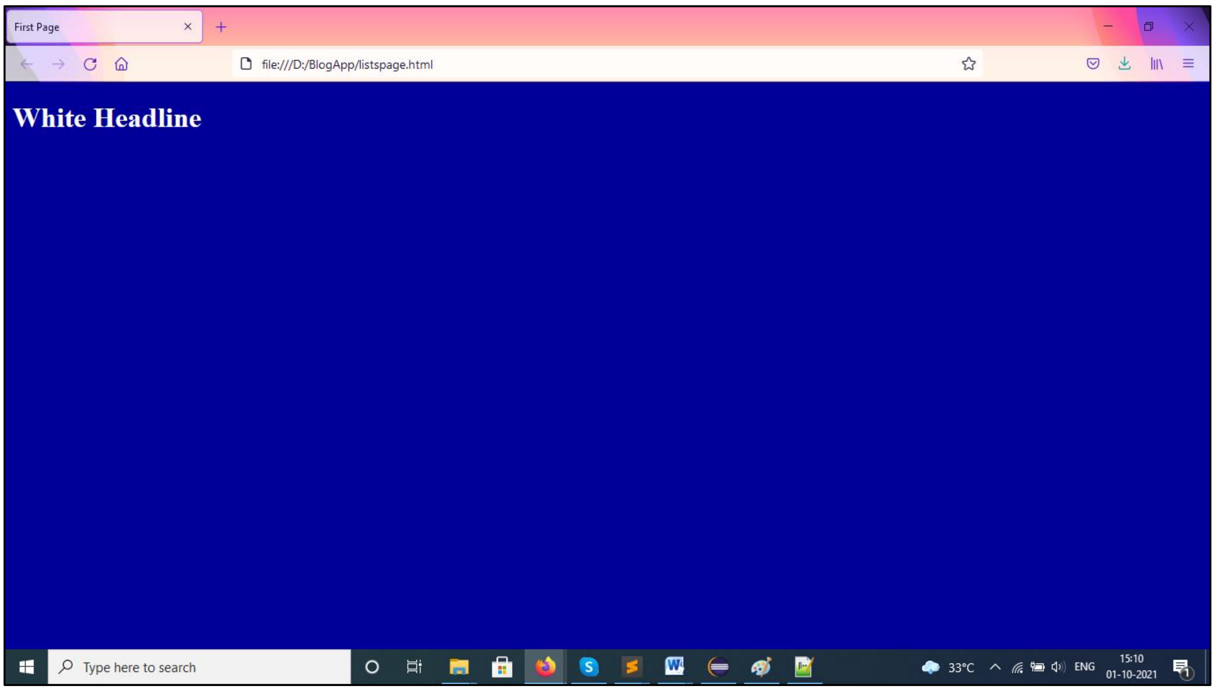
Background and Font Colors

The bgcolor attribute can be used to change the web page's background color.

This example would produce a dark blue background with a white headline:

```
<html>
<head>
<title>first page</title>
</head>
<body bgcolor="#000099">
<h1>
<font color="#FFFFFF"> White headline </font>
</h1>
</body>
</html>
```

Result:



The color attribute specifies the color of the text inside a <font> element.

HTML5

When writing HTML5 documents, one of the first new features that you'll notice is the doc type declaration:

```
<!DOCTYPE HTML>
```

The character encoding (charset) declaration is also simplified:

<meta charset="UTF-8">

New Elements in HTML5

`<article>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <footer>, <header>, <nav>, <output>, <progress>, <section>, <video>, and even more!`

The default character encoding in HTML5 is UTF-8.

New in HTML5

Forms

- ✓ The Web Forms 2.0 specification allows for creation of more powerful forms and more compelling user experiences.
- ✓ Date pickers, color pickers, and numeric stepper controls have been added.
- ✓ Input field types now include email, search, and URL.
- ✓ PUT and DELETE form methods are now supported.

Integrated API (Application Programming Interfaces)

- ✓ Drag and Drop
- ✓ Audio and Video
- ✓ Offline Web Applications
- ✓ History
- ✓ Local Storage
- ✓ Geolocation
- ✓ Web Messaging

The List of Content Models

In HTML, elements typically belonged in either the block level or inline content model. HTML5 introduces seven main content models.

- ✓ Metadata
- ✓ Embedded
- ✓ Interactive
- ✓ Heading
- ✓ Phrasing
- ✓ Flow
- ✓ Sectioning

The HTML5 content models are designed to make the markup structure more meaningful for both the browser and the web designer.

Content Models

Metadata

Content that sets up the presentation or behavior of the rest of the content. These elements are found in the head of the document.

Elements: `<base>, <link>, <meta>, <noscript>, <script>, <style>, <title>`

Embedded

Content that imports other resources into the document.

Elements: `<audio>, <video>, <canvas>, <iframe>, <img>, <math>, <object>, <svg>`

Interactive

Content specifically intended for user interaction.

Elements: `<a>, <audio>, <video>, <button>, <details>, <embed>, <iframe>, <img>, <input>, <label>, <object>, <select>, <textarea>`

Heading

Defines a section header.

Elements: `<h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <hgroup>`

Phrasing

This model has a number of inline level elements in common with HTML4.

Elements: `<img>, <span>, <strong>, <label>, <br />, <small>, <sub>`, and more.

The same element can belong to more than one content model.

Content Models

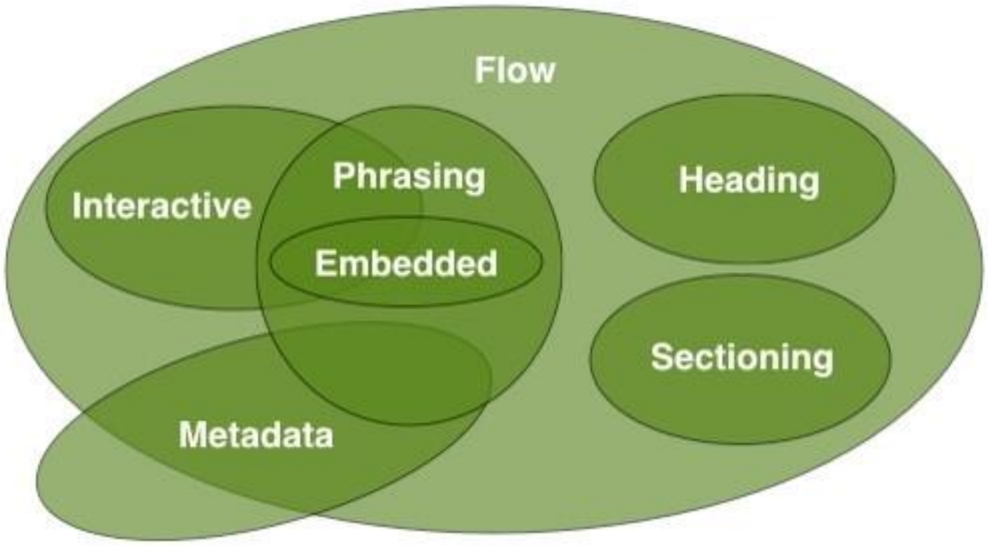
Flow content

Contains the majority of HTML5 elements that would be included in the normal flow of the document.

Sectioning content

Defines the scope of headings, content, navigation, and footers.

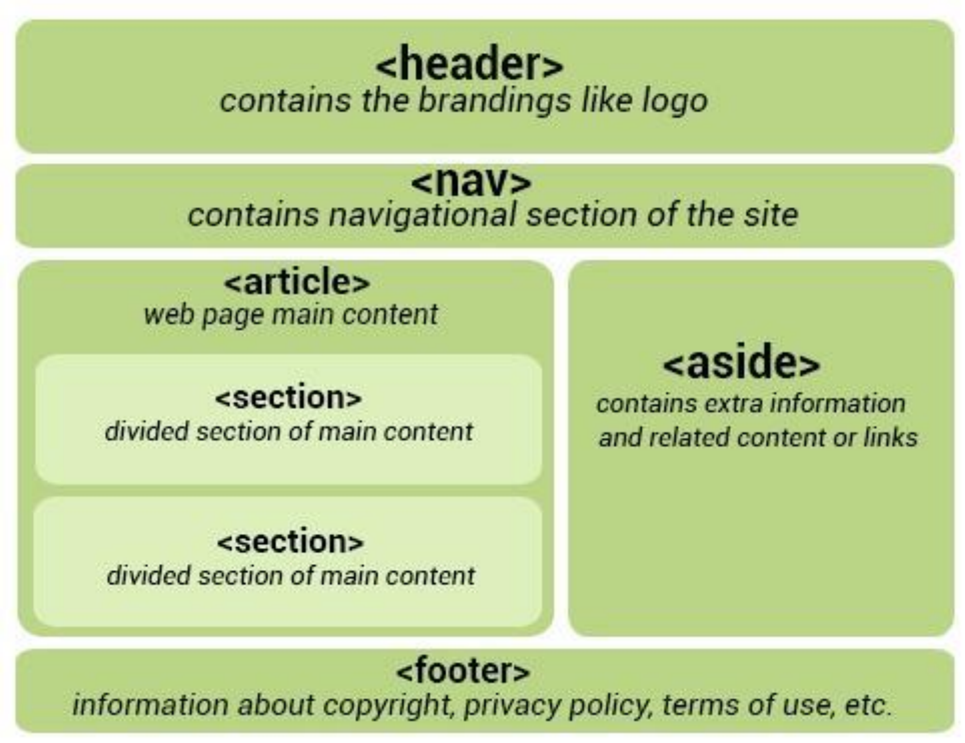
Elements: `<article>, <aside>, <nav>, <section>`



The various content models overlap in certain areas, depending on how they are being used.

Page Structure in HTML5

A generic HTML5 page structure looks like this:



<header>
contains the brandings like logo
<nav>
contains navigational section of the site
<article>
web page main content
<section>
divided section of main content
<section>
divided section of main content
<aside>
contains extra information and related content or links
<footer>
information about copyright, privacy policy, terms of use, etc.

You may not need some of these elements, depending on your page structure.

The <header> Element

In HTML4, we would define a header like this:

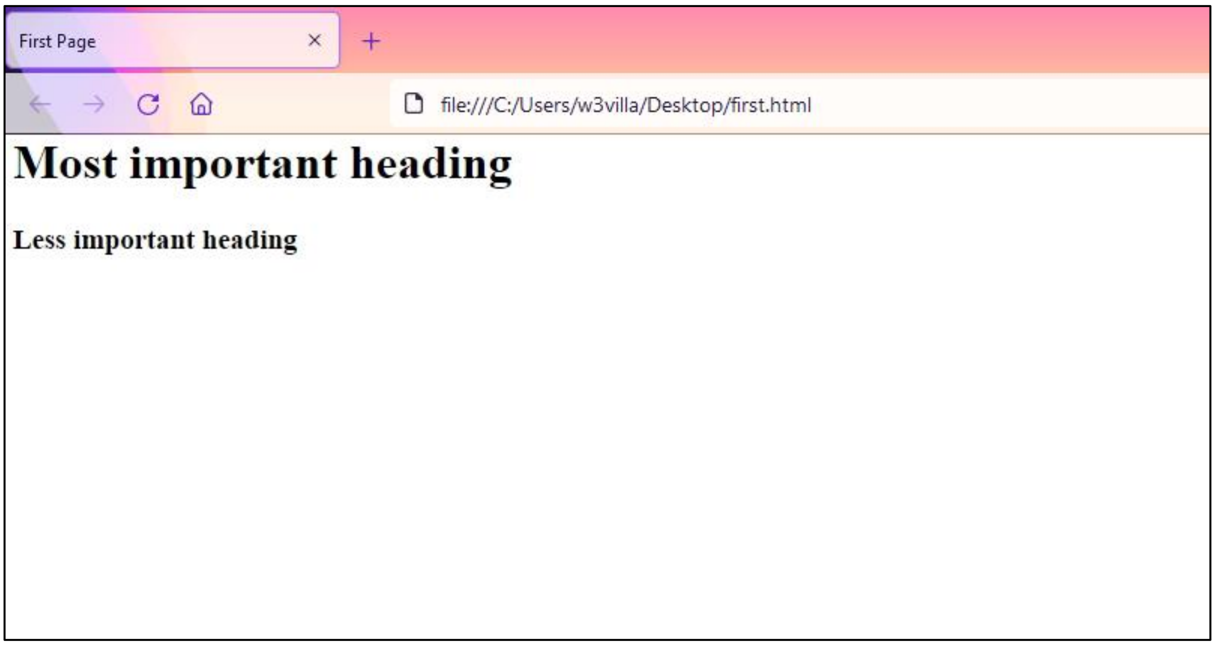
```
<div id="header">
```

In HTML5, a simple <header> tag is used, instead.

The <header> element is appropriate for use inside the body tag.

```
<!DOCTYPE html>
<html>
<head></head>
<body>
<header>
<h1> Most important heading </h1>
<h3> Less important heading </h3>
</header>
</body>
</html>
```

Result:



First Page

file:///C:/Users/w3villa/Desktop/first.html

Most important heading

Less important heading

Note that the <header> is completely different from the <head> tag.

The <footer> Element

The footer element is also widely used. Generally we refer to a section located at the very bottom of the web page as the footer.

```
<footer>…</footer>
```

The following information is usually provided between these tags:

- ✓ Contact Information
- ✓ Privacy Policy
- ✓ Social Media Icons
- ✓ Terms of Service
- ✓ Copyright Information
- ✓ Sitemap and Related Documents

The <nav> Element

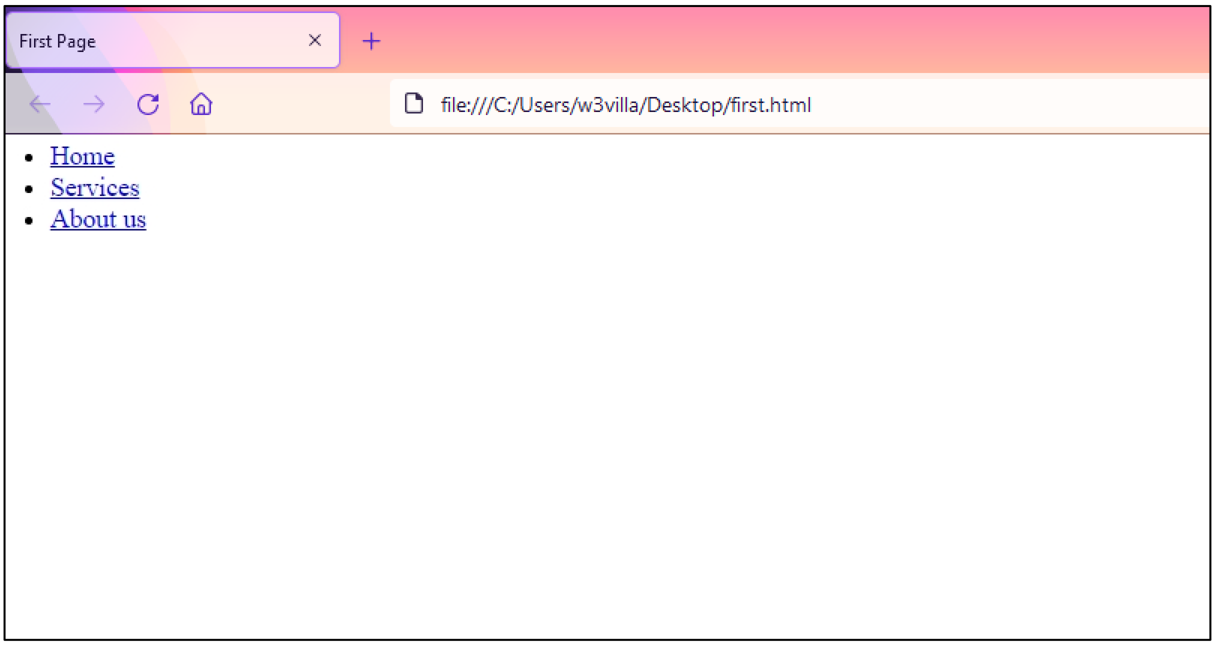
This tag represents a section of a page that links to other pages or to certain sections within the page.

This would be a section with navigation links.

Here is an example of a major block of navigation links:

```
<nav>
<ul>
<li><a href="#">Home</a></li>
<li><a href="#">Services</a></li>
<li><a href="#">About us</a></li>
</ul>
</nav>
```

Result:



- Home
- Services
- About us

Not all of the links in a document should be inside a <nav> element. The <nav> element is intended only for major blocks of navigation links. Typically, the <footer> element often has a list of links that don't need to be in a <nav> element.

The <article> Element

Article is a self-contained, independent piece of content that can be used and distributed separately from the rest of the page or site.

This could be a forum post, a magazine or newspaper article, a blog entry, a comment, an interactive widget or gadget, or any other independent piece of content.

The <article> element replaces the <div> element that was widely used in HTML4, along with an id or class.

```
<article>
<h1>The article title</h1>
<p>Contents of the article element </p>
</article>
```

When an <article> element is nested, the inner element represents an article related to the outer element. For example, blog post comments can be <article> elements nested in the <article> representing the blog post.

The <section> Element

<section> is a logical container of the page or article.

Sections can be used to divide up content within an article.

For example, a homepage could have a section for introducing the company, another for news items, and still another for contact information.

Each <section> should be identified, typically by including a heading (h1-h6 element) as a child of the <section> element.

```
<article>
<h1>Welcome</h1>
<section>
<h1>Heading</h1>
<p>content or image</p>
</section>
</article>
```

If it makes sense to separately syndicate the content of a <section> element, use an <article> element instead.

The <aside> Element

<aside> is secondary or tangential content which could be considered separate from but indirectly related to the main content.

This type of content is often represented in sidebars.

When an <aside> tag is used within an <article> tag, the content of the <aside> should be specifically related to that article.

```
<article>
<h1> Gifts for everyone </h1>
```

```
<p> This website will be the best place for choosing gifts
</p>
<aside>
<p> Gifts will be delivered to you within 24 hours </p>
</aside>
</article>
```

When an <aside> tag is used outside of an <article> tag, its content should be related to the surrounding content.

Audio on the Web

Before HTML5, there was no standard for playing audio files on a web page. The HTML5 <audio> element specifies a standard for embedding audio in a web page.

There are two different ways to specify the audio source file's URL. The first uses the source attribute:

```
<audio src="audio.mp3" controls>
Audio element not supported by your browser
</audio>
```

The second way uses the <source> element inside the <audio> element:

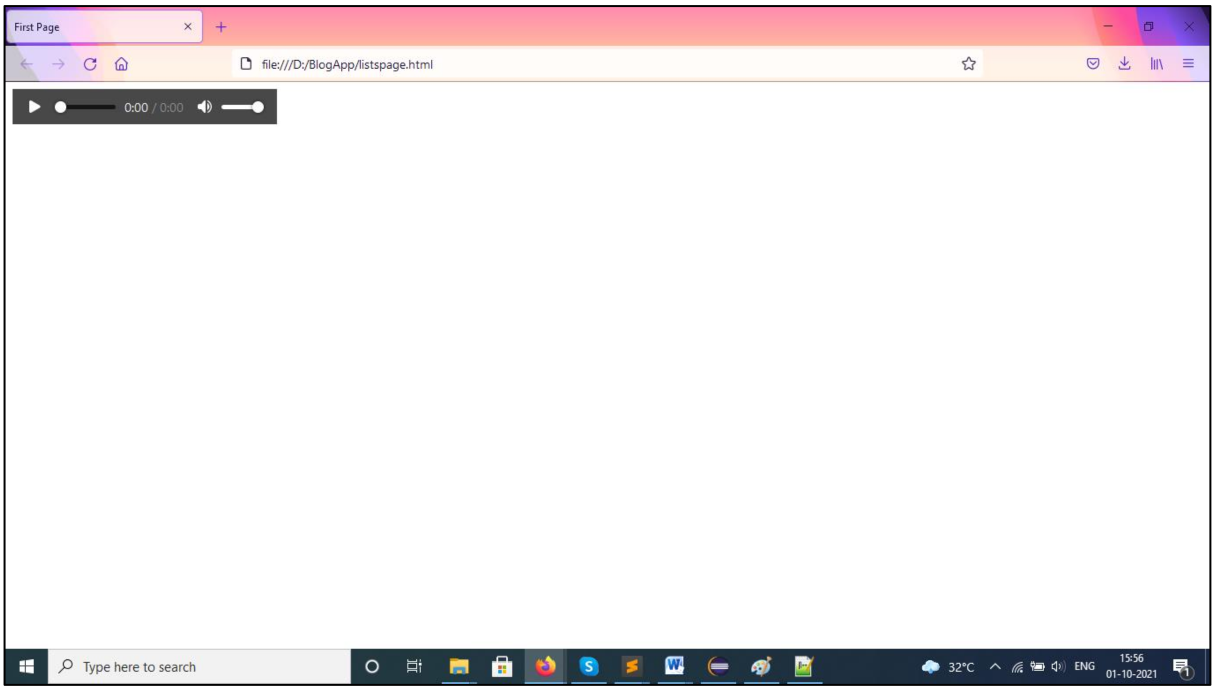
```
<audio controls>
<source src="audio.mp3" type="audio/mpeg">
<source src="audio.ogg" type="audio/ogg">
</audio>
```

Multiple <source> elements can be linked to different audio files. The browser will use the first recognized format.

The <audio> element creates an audio player inside the browser.

```
<audio controls>
<source src="audio.mp3" type="audio/mpeg">
<source src="audio.ogg" type="audio/ogg">
Audio element not supported by your browser.
</audio>
```

Result:



The text between the <audio> and </audio> tags will display in browsers that do not support the <audio> element.

Attributes of <audio>

controls

Specifies that audio controls should be displayed (such as a play/pause button, etc.)

autoplay

When this attribute is defined, audio starts playing as soon as it is ready, without asking for the visitor's permission.

```
<audio controls autoplay>
```

loop

This attribute is used to have the audio replay every time it is finished.

```
<audio controls autoplay loop>
```

Currently, there are three supported file formats for the <audio> element: MP3, WAV, and OGG.

Videos in HTML

The video element is similar to the audio element.

You can specify the video source URL using an attribute in a video element, or using source elements inside the video element:

```
<video controls>
<source src="video.mp4" type="video/mp4">
<source src="video.ogg" type="video/ogg">
Video is not supported by your browser
</video>
```

Another aspect that the audio and video elements have in common is that the major browsers do not all support the same file types. If the browser does not support the first video type, it will try the next one.

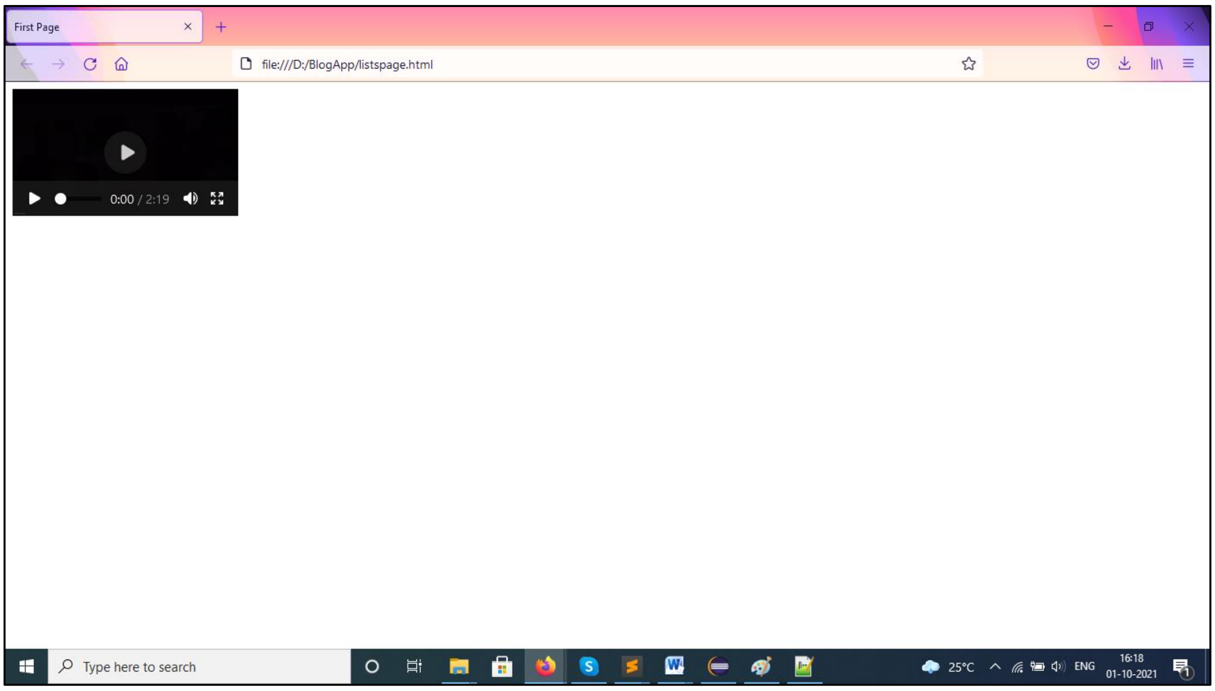
Attributes of <video>

Another aspect shared by both the audio and the video elements is that each has controls, autoplay and loop attributes.

In this example, the video will replay after it finishes playing:

```
<video controls autoplay loop>
<source src="video.mp4" type="video/mp4">
<source src="video.ogg" type="video/ogg">
Video is not supported by your browser
</video>
```

Result:



Currently, there are three supported video formats for the <video> element: MP4, WebM, and OGG.

Progress Bar

The <progress> element provides the ability to create progress bars on the web.

The progress element can be used within headings, paragraphs, or anywhere else in the body.

Progress Element Attributes

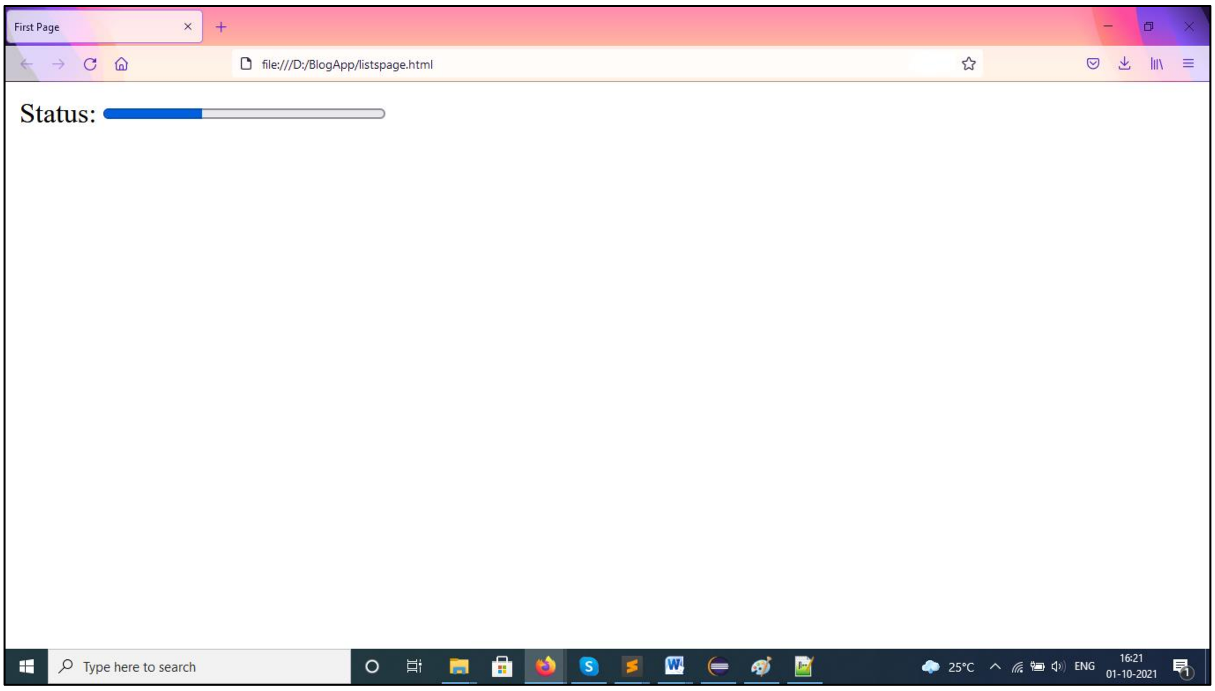
Value: Specifies how much of the task has been completed.
Max: Specifies how much work the task requires in total.

Example:

Status: `<progress min="0" max="100" value="35">`

```
</progress>
```

Result:



Status:

Use the <progress> tag in conjunction with JavaScript to dynamically display a task's progress.

HTML5 Web Storage

With HTML5 web storage, websites can store data on a user's local computer.

Before HTML5, we had to use JavaScript cookies to achieve this functionality.

The Advantages of Web Storage

- ✓ More secure
- ✓ Faster
- ✓ Stores a larger amount of data
- ✓ Stored data is not sent with every server request

Local storage is per domain. All pages from one domain can store and access the same data.

Types of Web Storage Objects

There are two types of web storage objects:

- ✓ sessionStorage()
- ✓ localStorage()

Local vs. Session

- ✓ Session Storage is destroyed once the user closes the browser
- ✓ Local Storage stores data with no expiration date

You need to be familiar with basic JavaScript in order to understand and use the API.

Working with Values

The syntax for web storage for both local and session storage is very simple and similar.

The data is stored as key/value pairs.

Storing a Value:

```
localStorage.setItem("key1", "value1");
```

Getting a Value:

```
//this will print the value
alert(localStorage.getItem("key1"));
```

Removing a Value:

```
localStorage.removeItem("key1");
```

Removing All Values:

```
localStorage.clear();
```

The same syntax applies to the session storage, with one difference: Instead of localStorage, sessionStorage is used.

What is the Geolocation API?

In HTML5, the Geolocation API is used to obtain the user's geographical location.

Since this can compromise user privacy, the option is not available unless the user approves it.

Geolocation is much more accurate for devices with GPS, like smartphones and the like.

Using HTML Geolocation

The Geolocation API's main method is getCurrentPosition, which retrieves the current geographic location of the user's device.

navigator.geolocation.getCurrentPosition();

Parameters:

showLocation (mandatory): Defines the callback method that retrieves location information.

ErrorHandler(optional): Defines the callback method that is invoked when an error occurs in processing the asynchronous call.

Options (optional): Defines a set of options for retrieving the location information.

You need to be familiar with basic JavaScript in order to understand and use the API.

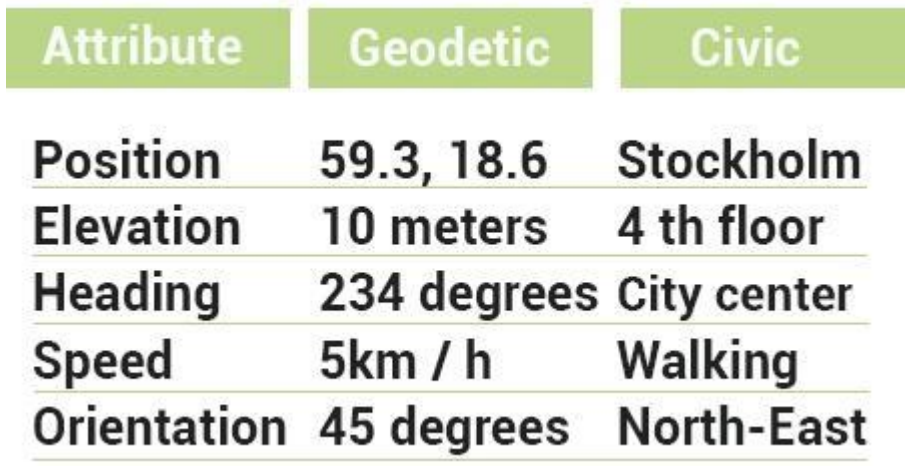
Presenting Data

User location can be presented in two ways: Geodetic and Civic.

1. The geodetic way to describe position refers directly to latitude and longitude.

2. The civic representation of location data is presented in a format that is more easily read and understood by the average person.

Each parameter has both a geodetic and a civic representation:



Attribute	Geodetic	Civic
Position	59.3, 18.6	Stockholm
Elevation	10 meters	4 th floor
Heading	234 degrees	City center
Speed	5km / h	Walking
Orientation	45 degrees	North-East

The getCurrentPosition() method returns an object if it is successful. The latitude, longitude, and accuracy properties are always returned.

Making Elements Draggable

The drag and drop feature lets you "grab" an object and drag it to a different location.
To make an element draggable, just set the draggable attribute to true:

```
<img draggable="true" />
```

Any HTML element can be draggable. The API for HTML5 drag and drop is event-based.

Example:

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
ev.preventDefault();
}

function drag(ev) {
ev.dataTransfer.setData("text", ev.target.id);
}
```

```
function drop(ev) {
ev.preventDefault();
var data = ev.dataTransfer.getData("text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="box" ondrop="drop(event)"
ondragover="allowDrop(event)"
style="border:1px solid black;
width:200px; height:200px"></div>

<img id="image" src="sample.jpg" draggable="true"
ondragstart="drag(event)" width="150" height="50" alt=""
/>

</body>
</html>
```

What to Drag

When the element is dragged, the ondragstart attribute calls a function, drag (event), which specifies what data is to be dragged.

The dataTransfer.setData () method sets the data type and the value of the dragged data:

```
function drag(ev) {
ev.dataTransfer.setData("text", ev.target.id);
}
```

In our example, the data type is "text" and the value is the ID of the draggable element ("image").

Where to Drop

The ondragover event specifies where the dragged data can be dropped. By default, data and elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.

This is done by calling the event.preventDefault() method for the ondragover event.

Do the Drop

When the dragged data is dropped, a drop event occurs.
In the example above, the ondrop attribute calls a function, drop(event):

```
function drop(ev) {
ev.preventDefault();
var data = ev.dataTransfer.getData("text");
ev.target.appendChild(document.getElementById(data));
}
```

The preventDefault() method prevents the browser's default handling of the data (default is open as link on drop).

The dragged data can be accessed with the dataTransfer.getData() method.

This method will return any data that was set to the same type in the setData() method.

The dragged data is the ID of the dragged element ("image").

At the end, the dragged element is appended into the drop element, using the appendChild() function.

Basic knowledge of JavaScript is required to understand and use the API.

Drawing Shapes

SVG stands for Scalable Vector Graphics, and is used to draw shapes with HTML-style markup.

It offers several methods for drawing paths, boxes, circles, text, and graphic images.

SVG is not pixel-based, so it can be magnified infinitely with no loss of quality.
Inserting SVG Images

An SVG image can be added to HTML code with just a basic image tag that includes a source attribute pointing to the image:

```
<img src="image.svg" alt="" height="300" />
```

SVG defines vector-based graphics in XML format.
Drawing a Circle

To draw shapes with SVG, you first need to create an SVG element tag with two attributes: width and height.

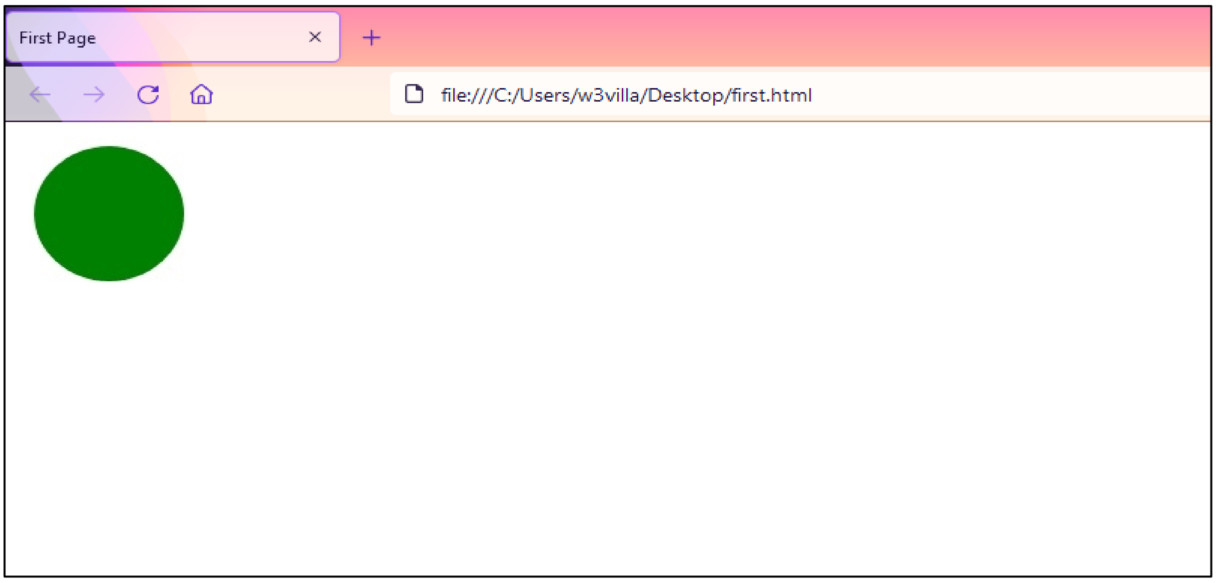
```
<svg width="1000" height="1000"></svg>
```

To create a circle, add a <circle> tag:

```
<svg width="2000" height="2000">
<circle cx="80" cy="80" r="50" fill="green" />
</svg>
```

- ✓ cx pushes the center of the circle further to the right of the screen
- ✓ cy pushes the center of the circle further down from the top of the screen
- ✓ r defines the radius
- ✓ fill determines the color of our circle
- ✓ stroke adds an outline to the circle

Result:



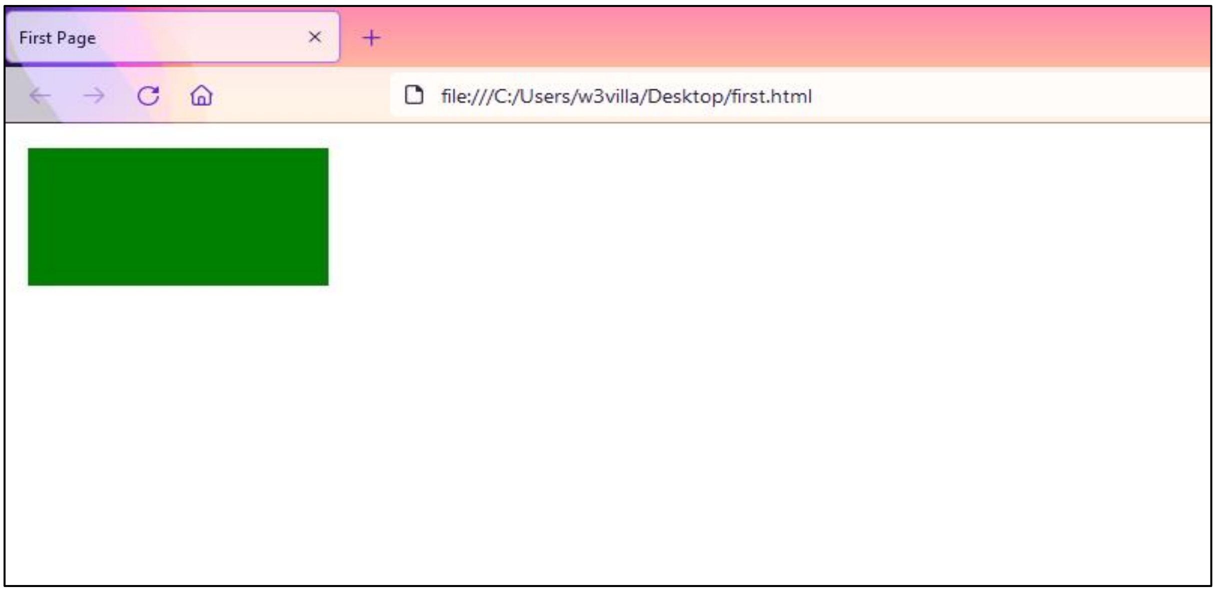
Every element and every attribute in SVG files can be animated.

Other Shape Elements

```
<rect> defines a rectangle:
<svg width="2000" height="2000">
<rect width="300" height="100"
x="20" y="20" fill="green" />
</svg>
```

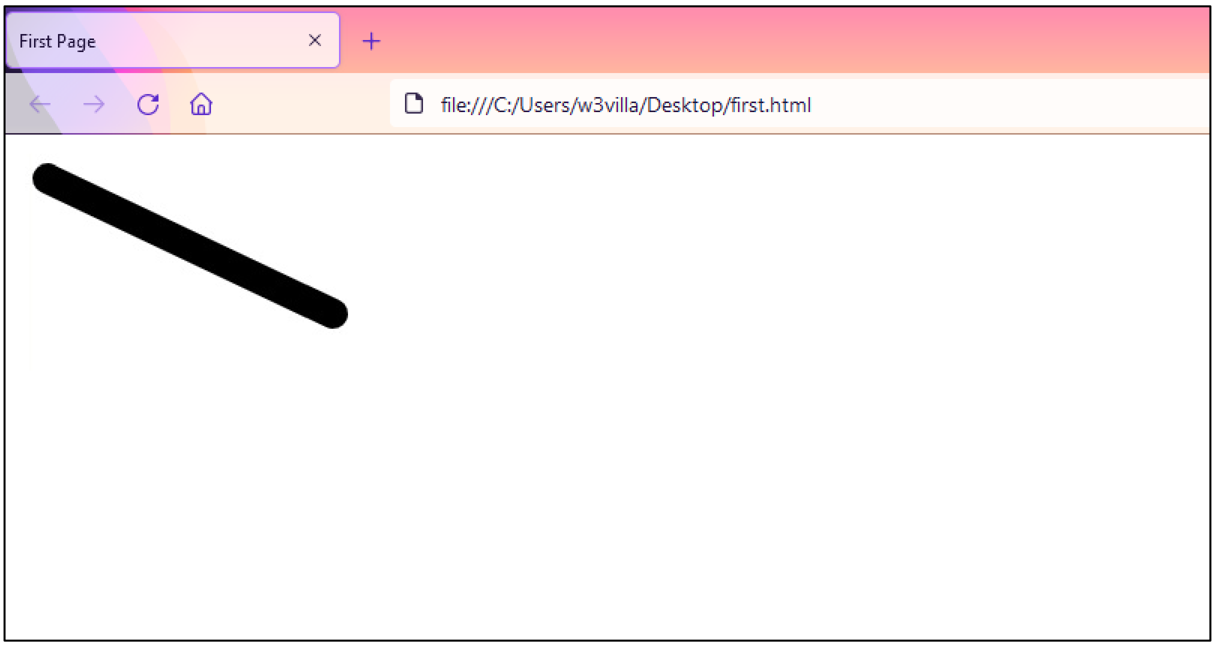
The following code will draw a green-filled rectangle.

Result:



```
<line> defines a line segment:
<svg width="400" height="410">
<line x1="10" y1="10" x2="200" y2="100"
style="stroke:#000000; stroke-linecap:round;
stroke-width:20" />
</svg>
```

(x1, y1) define the start coordinates (x2, y2) define the end coordinates.

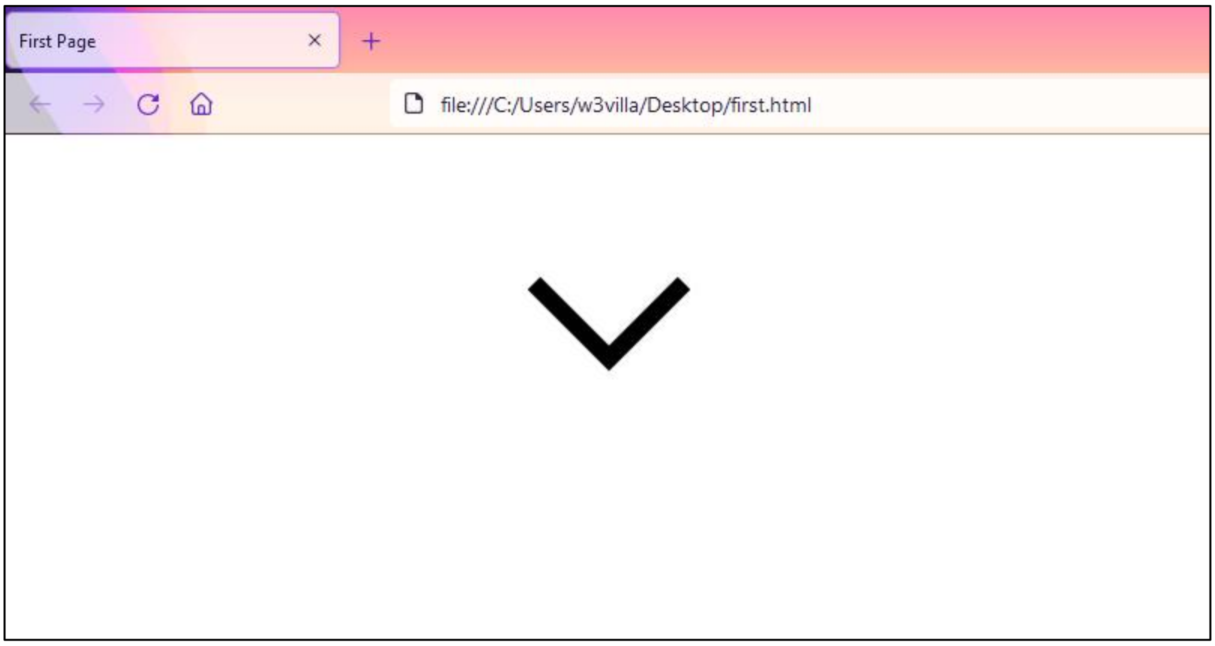


<polyline> defines shapes built from multiple line definitions:

```
<svg width="2000" height="500">
<polyline style="stroke-linejoin:miter; stroke:black;
stroke-width:12; fill: none;"
points="100 100, 150 150, 200 100" />
</svg>
```

Points are the polyline's coordinates.

The code below will draw a black check sign:



The width and height attributes of the <rect> element define the height and the width of the rectangle.

<ellipse> and <polygon>

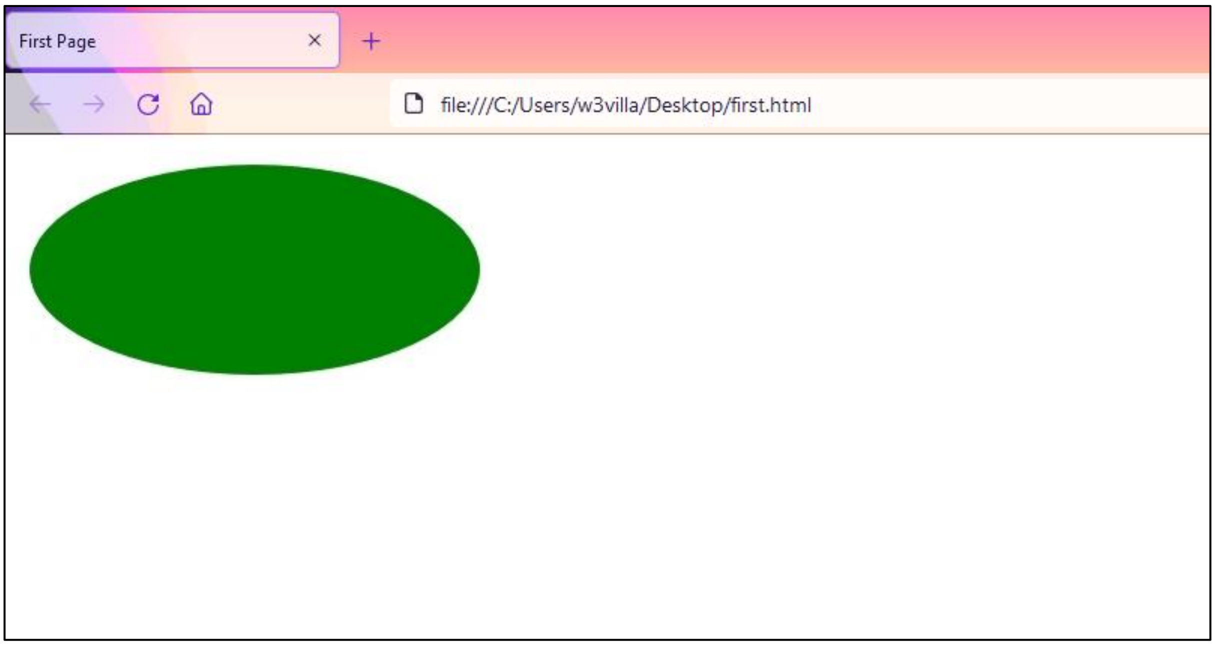
Ellipse

The <ellipse> is similar to the <circle>, with one exception:

You can independently change the horizontal and vertical axes of its radius, using the rx and ry attributes.

```
<svg height="500" width="1000">
<ellipse cx="200" cy="100" rx="150" ry="70"
style="fill:green" />
</svg>
```

Result:

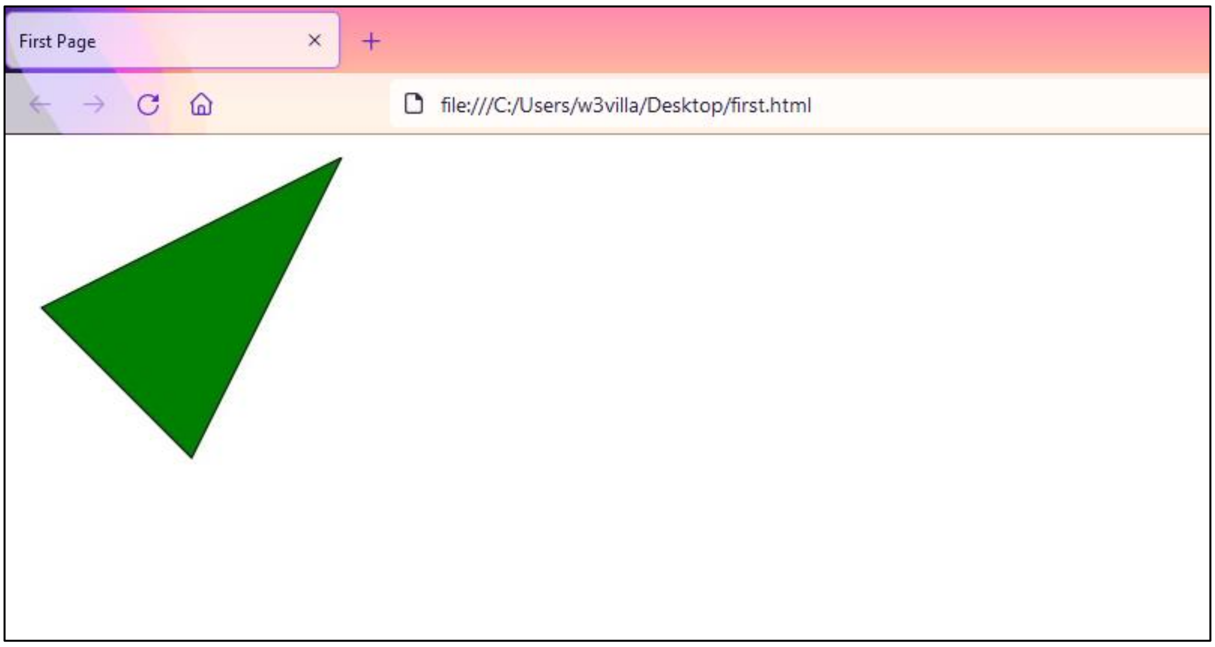


Polygon

The <polygon> element is used to create a graphic with at least three sides. The polygon element is unique because it automatically closes off the shape for you.

```
<svg width="2000" height="2000">
<polygon points="100 100, 200 200, 300 0"
style="fill: green; stroke:black;" />
</svg>
```

Result:



Polygon comes from Greek. "Poly" means "many" and "gon" means "angle."

Shape Animations

SVG animations can be created using the <animate> element.

The example below creates a rectangle that will change its position in 3 seconds and will then repeat the animation twice:

```
<svg width="1000" height="250">
<rect width="150" height="150" fill="orange">
<animate attributeName="x" from="0" to="300"
dur="3s" fill="freeze" repeatCount="2"/>
</rect>
</svg>
```

attributeName: Specifies which attribute will be affected by the animation

from: Specifies the starting value of the attribute

to: Specifies the ending value of the attribute

dur: Specifies how long the animation runs (duration)

fill: Specifies whether or not the attribute's value should return to its initial value when the animation is finished (Values: "remove" resets the value; "freeze" keeps the "to value")

repeatCount: Specifies the repeat count of the animation

In the example above, the rectangle changes its x attribute from 0 to 300 in 3 seconds.

To repeat the animation indefinitely, use the value "indefinite" for the repeatCount attribute.

Paths

The <path> element is used to define a path.

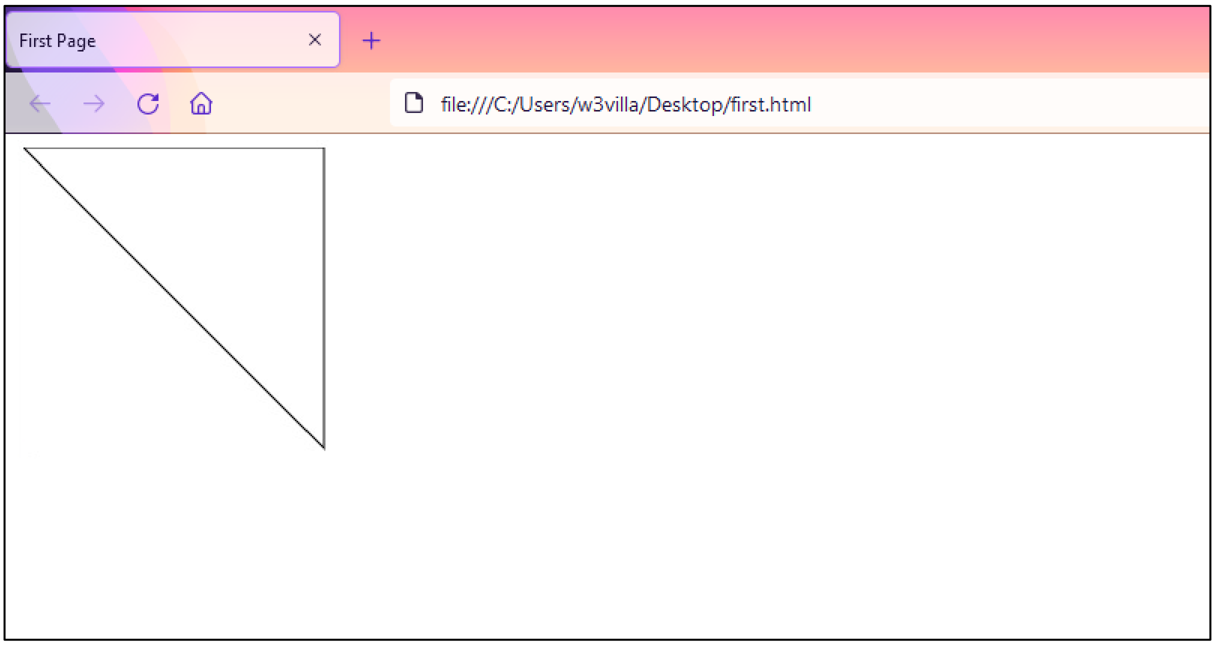
The following commands are available for path data:

M: moveto
L: lineto
H: horizontal lineto
V: vertical lineto
C: curveto
S: smooth curveto
Q: quadratic Bézier curve
T: smooth quadratic Bézier curveto
A: elliptical Arc
Z: closepath

Define a path using the d attribute:

```
<svg width="500" height="500">
<path d="M 0 0 L200 200 L200 0 Z" style="stroke:#000;
fill:none;" />
</svg>
```

M places our "virtual pen" down at the position 0, 0. It then moves 200px down and to the right, then moves up to the position 200, 0. The Z command closes the shape, which results in a hypotenuse:



All of the above commands can also be expressed with lower case letters. When capital letters are used, it indicates absolute position; lower case indicates relative position.

The <canvas> Element

The HTML canvas is used to draw graphics that include everything from simple lines to complex graphic objects.

The <canvas> element is defined by:

```
<canvas id="canvas1" width="200" height="100">
</canvas>
```

The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics (usually JavaScript).

The <canvas> element must have an id attribute so it can be referred to by JavaScript:

```
<html>
<head></head>
<body>
<canvas id="canvas1"
width="400" height="300"></canvas>
<script>
var can = document.getElementById("canvas1");
var ctx = can.getContext("2d");
</script>
</body>
</html>
```

getContext() returns a drawing context on the canvas.

Basic knowledge of JavaScript is required to understand and use the Canvas.

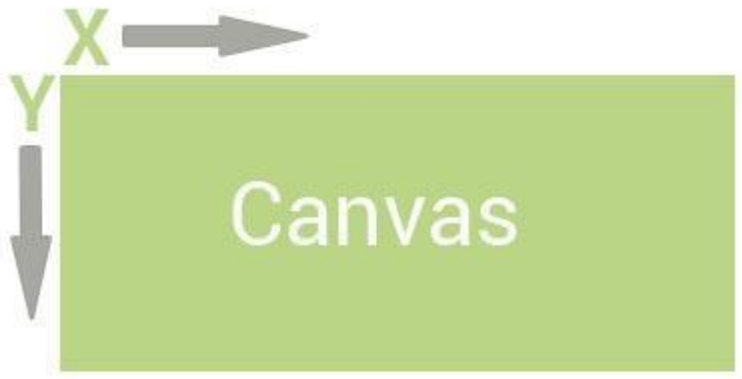
Canvas Coordinates

The HTML canvas is a two-dimensional grid.

The upper-left corner of the canvas has the coordinates (0,0).

X coordinate increases to the right.

Y coordinate increases toward the bottom of the canvas.



The <canvas> element is only a container for graphics.

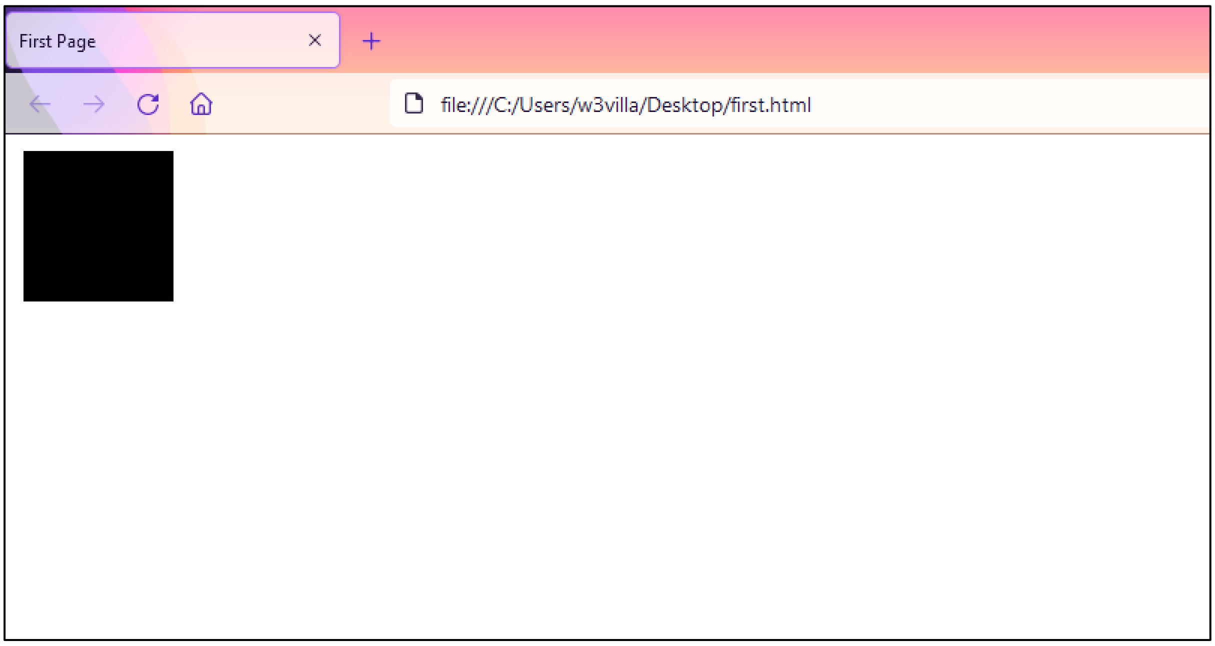
Drawing Shapes

The fillRect(x, y, w, h) method draws a "filled" rectangle, in which w indicates width and h indicates height. The default fill color is black.

A black 100*100 pixel rectangle is drawn on the canvas at the position (20, 20):

```
var c=document.getElementById("canvas1");
var ctx=c.getContext("2d");
ctx.fillRect(20,20,100,100);
```

Result:

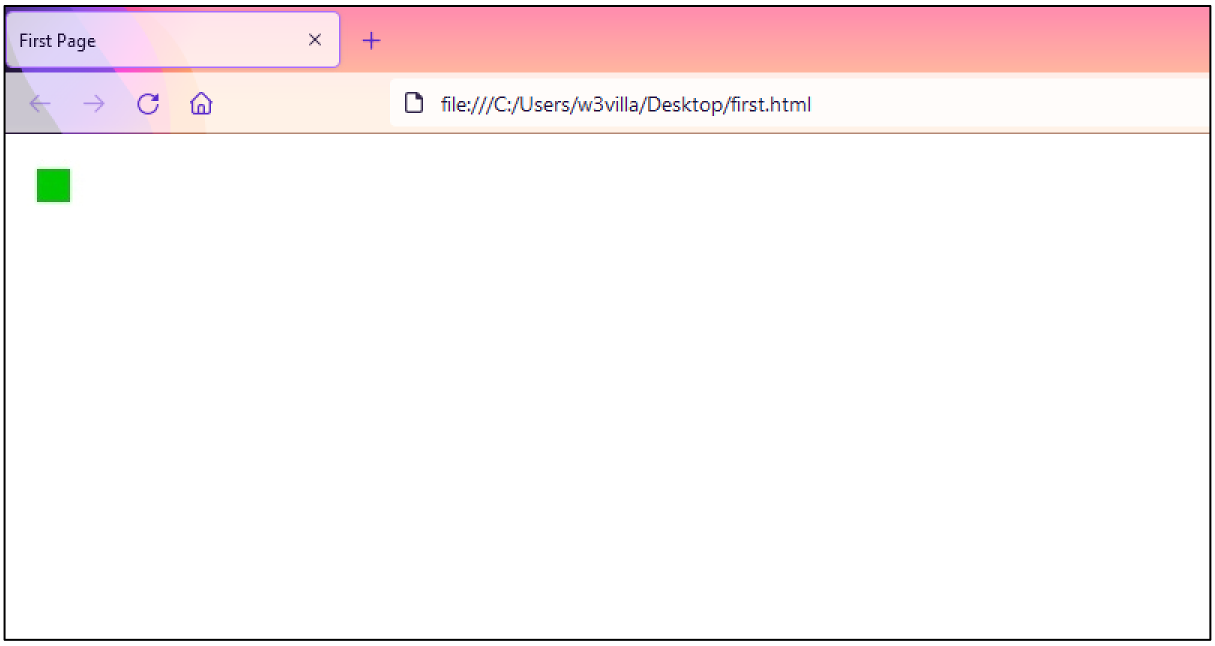


The fillStyle property is used to set a color, gradient, or pattern to fill the drawing.

Using this property allows you to draw a green-filled rectangle.

```
var canvas=document.getElementById("canvas1");
var ctx=canvas.getContext("2d");
ctx.fillStyle ="rgba(0, 200, 0, 1)";
ctx.fillRect (36, 10, 22, 22);
```

Result:



The canvas supports various other methods for drawing:

Draw a Line

moveTo(x,y): Defines the starting point of the line.
lineTo(x,y): Defines the ending point of the line.

Draw a Circle

beginPath(): Starts the drawing.
arc(x,y,r,start,stop): Sets the parameters of the circle.
stroke(): Ends the drawing.

Gradients

createLinearGradient(x,y,x1,y1): Creates a linear gradient.
createRadialGradient(x,y,r,x1,y1,r1): Creates a radial/circular gradient.

Drawing Text on the Canvas

Font: Defines the font properties for the text.
fillText(text,x,y): Draws "filled" text on the canvas.
strokeText(text,x,y): Draws text on the canvas (no fill).

There are many other methods aimed at helping to draw shapes and images on the canvas.

Canvas vs. SVG

Canvas

- ✓ Elements are drawn programmatically
- ✓ Drawing is done with pixels
- ✓ Animations are not built in
- ✓ High performance for pixels-based drawing operations
- ✓ Resolution dependent
- ✓ No support for event handlers
- ✓ You can save the resulting image as .png or .jpg
- ✓ Well suited for graphic-intensive games

SVG

- ✓ Elements are part of the page's DOM (Document object model)
- ✓ Drawing is done with vectors
- ✓ Effects, such as animations are built in
- ✓ Based on standard XML syntax, which provides better accessibility
- ✓ Resolution independent
- ✓ Support for event handlers
- ✓ Not suited for game applications
- ✓ Best suited for applications with large rendering areas (for example, Google Maps)

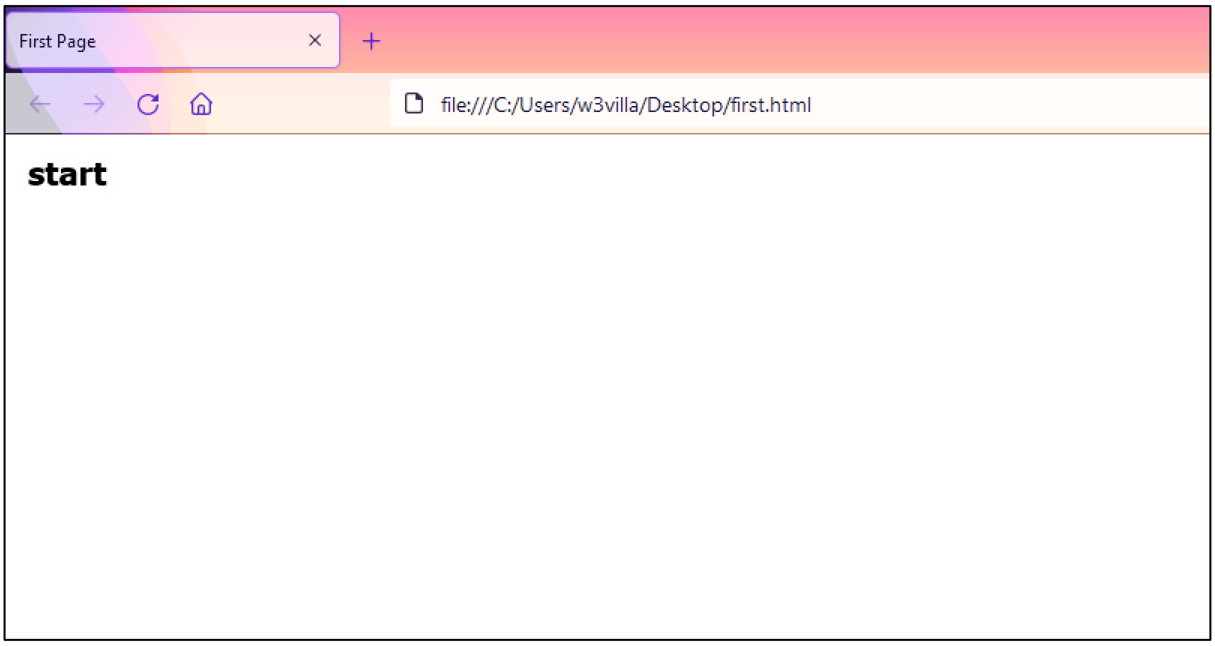
You can actually use both SVG and canvas on the same page, if needed. However, you cannot just draw SVG onto a canvas, or vice-versa.

Working with Canvas

The Canvas element can be transformed. As an example, a text is written on the canvas at the coordinates (10, 30).

```
ctx.font="bold 22px Tahoma";
ctx.textAlign="start";
ctx.fillText("start", 10, 30);
```

Result:



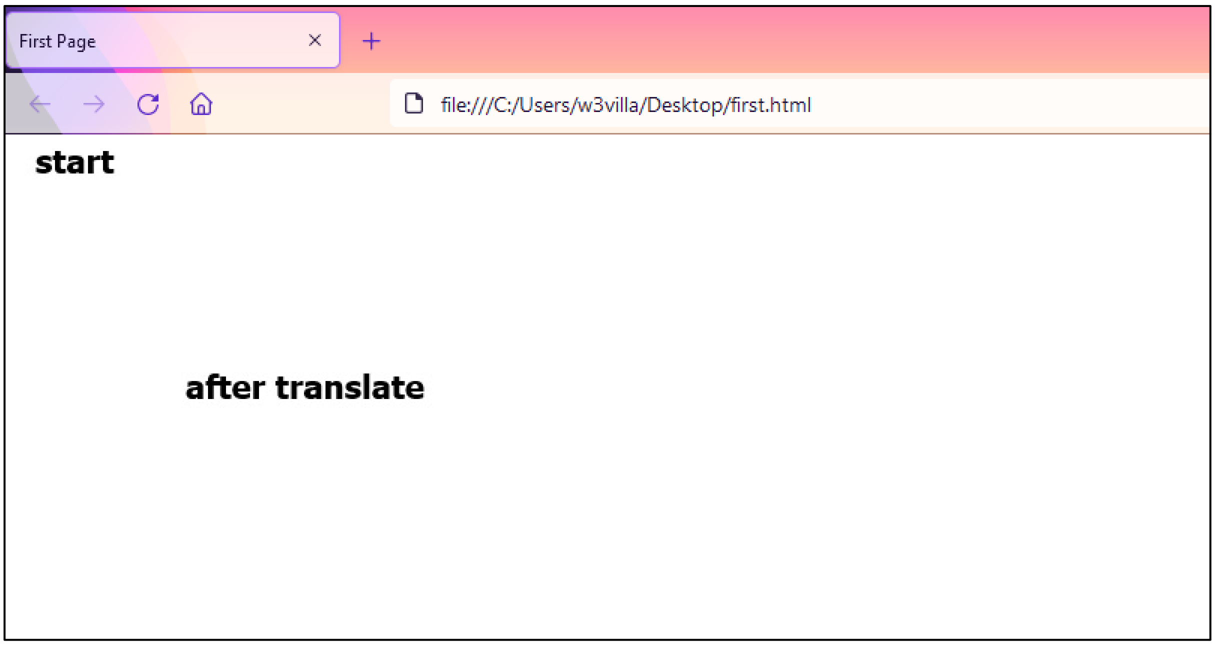
The translate(x,y) method is used to move the Canvas.

x indicates how far to move the grid horizontally, and y indicates how far to move the grid vertically.

```
.ctx.translate(100, 150);
ctx.fillText("after translate", 10, 30);
```

In this example, the canvas is moved 100px to the right, and 150px down.

Result:



start

after translate

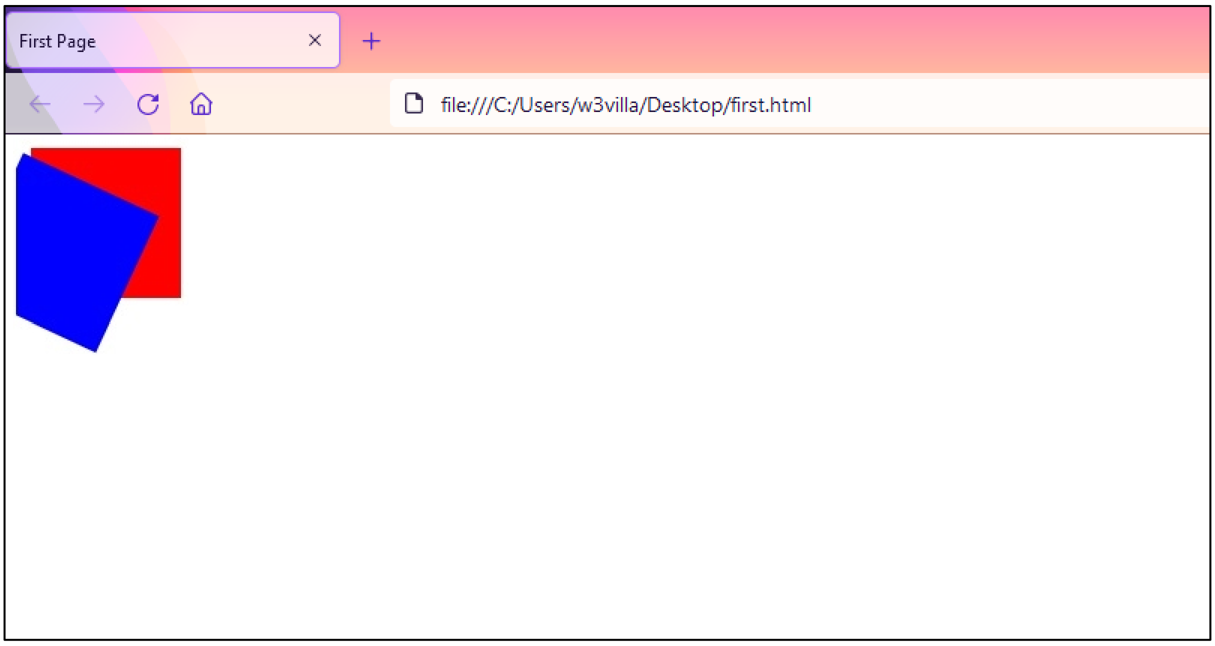
Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

The rotate() Method

The rotate() method is used to rotate the HTML5 Canvas. The value must be in radians, not degrees.

Here is an example that draws the same rectangle before and after rotation is set:

```
ctx.fillStyle = "#FF0000";
ctx.fillRect(10,10, 100, 100);
ctx.rotate( (Math.PI / 180) * 25); //rotate 25 degrees.
ctx.fillStyle = "#0000FF";
ctx.fillRect(10,10, 100, 100);
```



The rotation will only affect drawings made after the rotation is done.

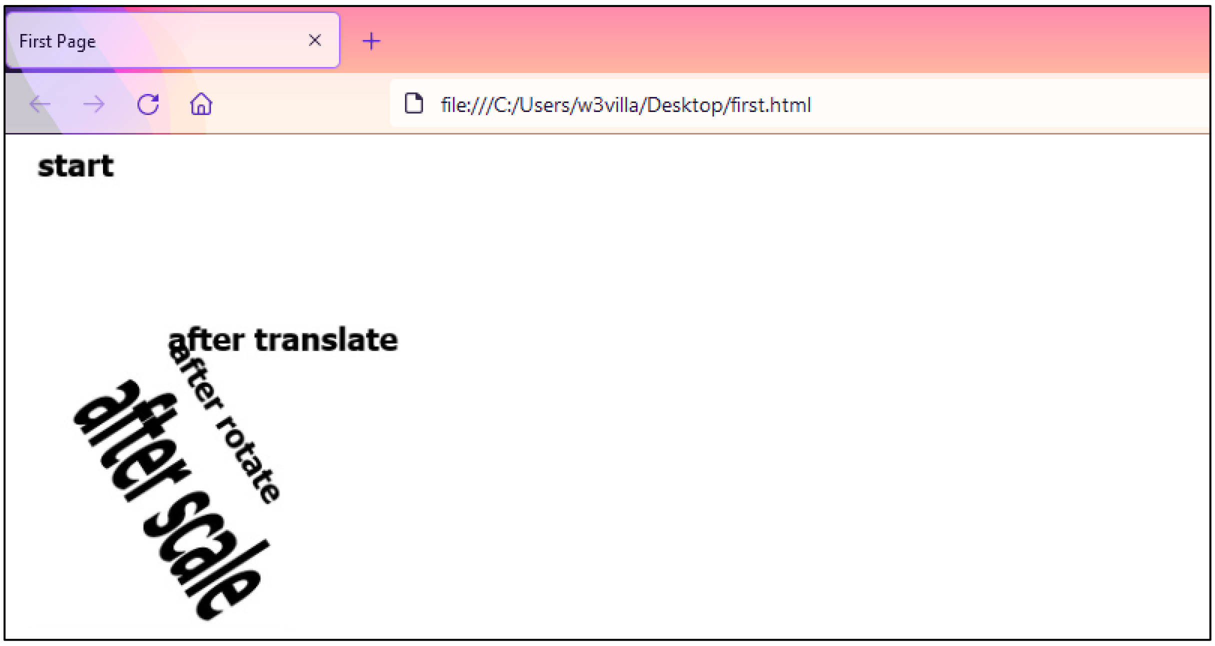
The scale() Method

The scale() method scales the current drawing. It takes two parameters:

- ✓ The number of times by which the image should be scaled in the X-direction.
- ✓ The number of times by which the image should be scaled in the Y-direction.

```
var canvas = document.getElementById('canvas1');
ctx =canvas.getContext('2d');
ctx.font="bold 22px Tahoma";
ctx.textAlign="start";
ctx.fillText("start", 10, 30);
ctx.translate(100, 150);
ctx.fillText("after translate", 0, 0);
ctx.rotate(1);
ctx.fillText("after rotate", 0, 0);
ctx.scale(1.5, 4);
ctx.fillText("after scale", 0,20);
```

This will scale the canvas 1.5 times in the X-direction, and 4 times in Y-direction:



If you scale a drawing, all future drawings will also be scaled.

HTML5 Forms

HTML5 brings many features and improvements to web form creation. There are new attributes and input types that were introduced to help create better experiences for web users.

Form creation is done in HTML5 the same way as it was in HTML4:

```
<form>
<label>Your name:</label>
<input id="user" name="username" type="text" />
</form>
```

Use the novalidate attribute to avoid form validation on submissions.

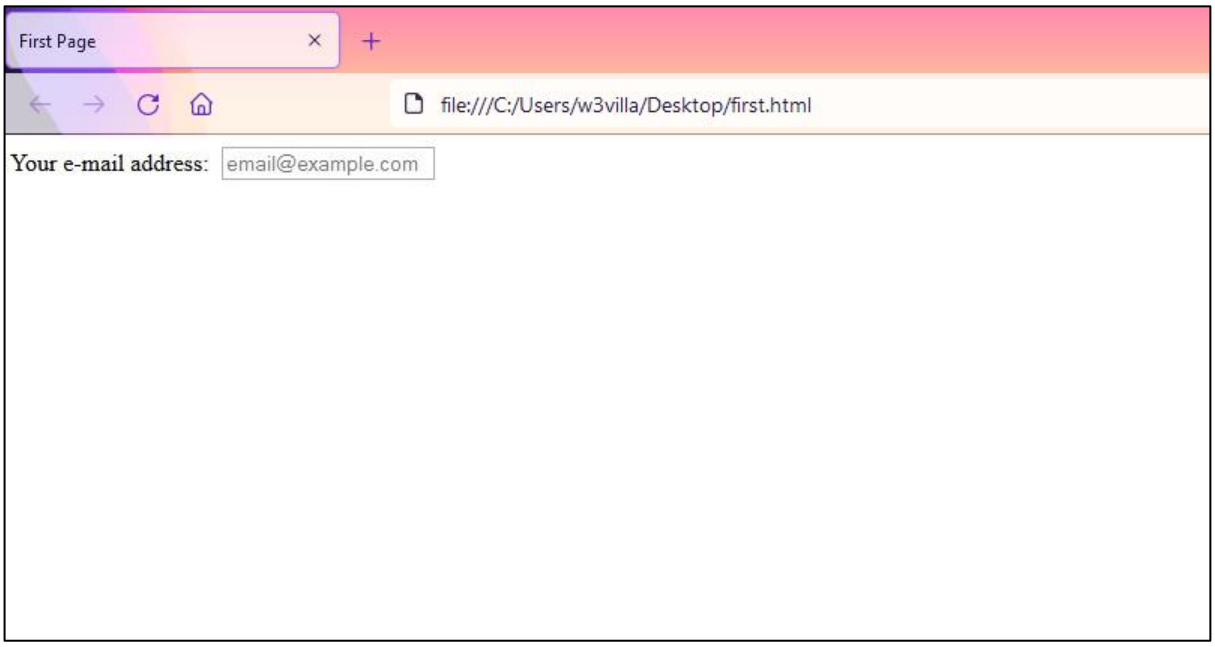
New Attributes

HTML5 has introduced a new attribute called placeholder. On <input> and <textarea> elements, this attribute provides a hint to the user of what information can be entered into the field.

```
<form>
<label for="email">Your e-mail address: </label>
<input type="text" name="email"
placeholder="email@example.com" />
```

```
</form>
```

Result:

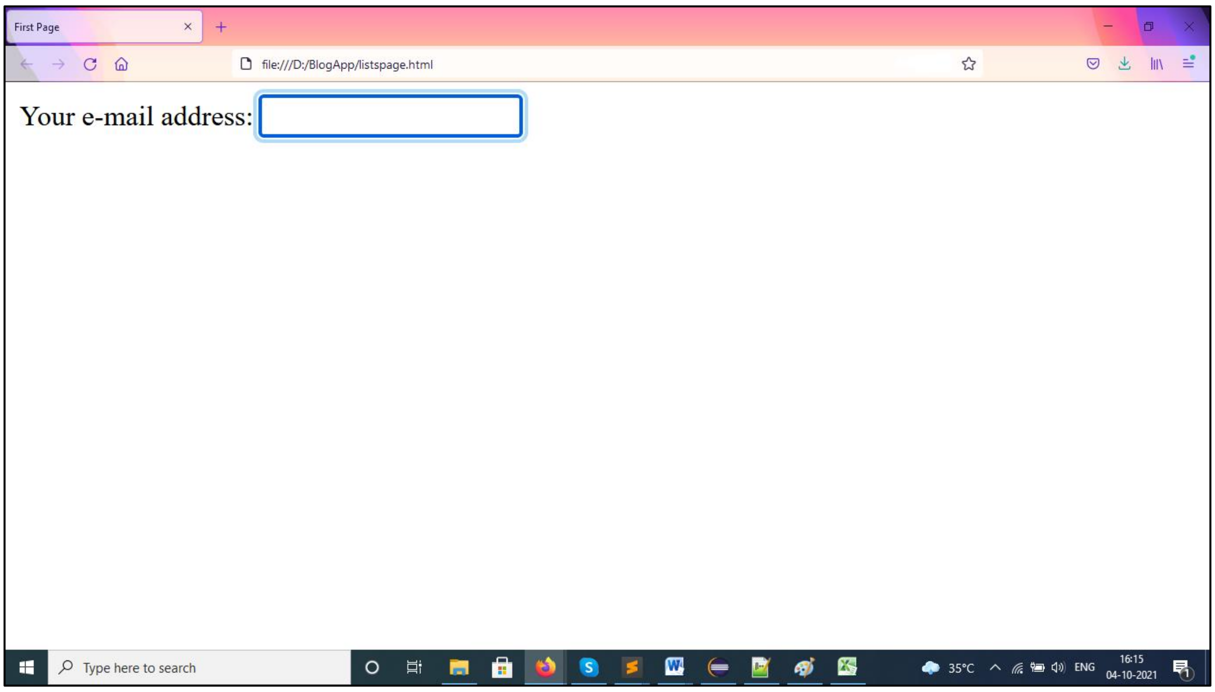


Your e-mail address: email@example.com

The autofocus attribute makes the desired input focus when the form loads:

```
<form>
<label for="e-mail">Your e-mail address: </label>
<input type="text" name="email" autofocus/>
</form>
```

Result:



Your e-mail address:

The required attribute tells the browser that the input is required.

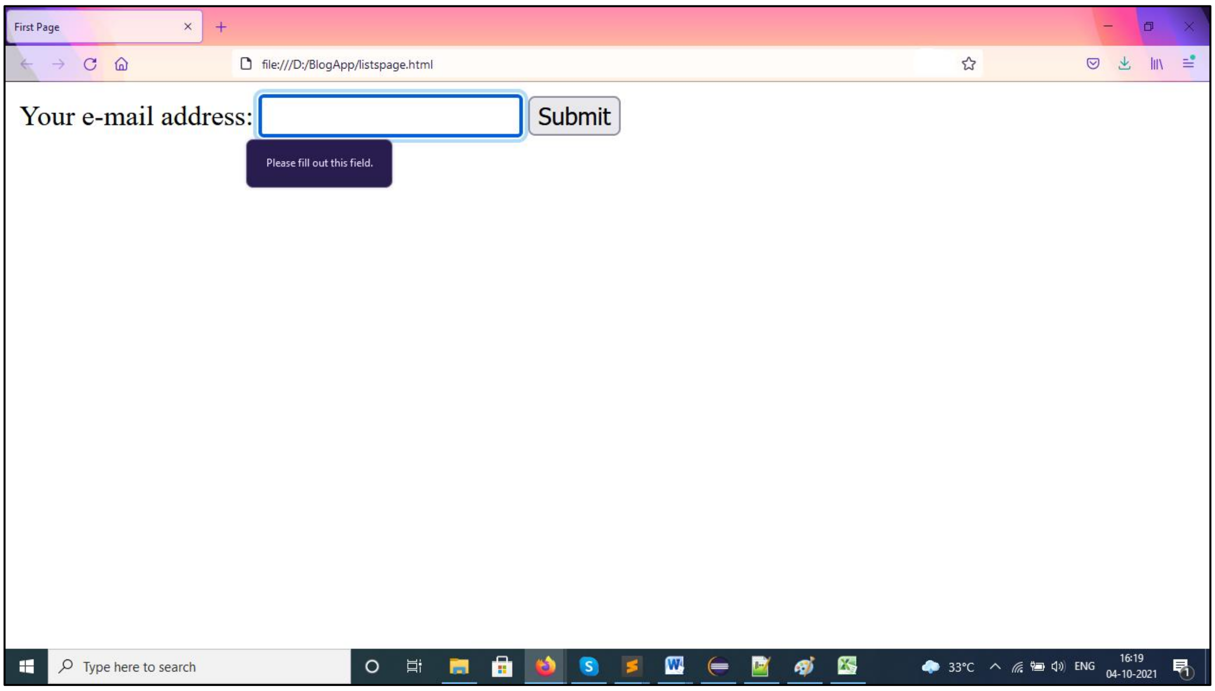
Forms with Required Fields

The "required" attribute is used to make the input elements required.

```
<form autocomplete="off">
<label for="e-mail">Your e-mail address: </label>
<input name="Email" type="text" required />
<input type="submit" value="Submit"/>
</form>
```

The form will not be submitted without filling in the required fields.

Result:



Your e-mail address: Submit

Please fill out this field.

The autocomplete attribute specifies whether a form or input field should have autocomplete turned on or off.

When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

HTML5 added several new input types:

- ✓ color
- ✓ date

- ✓ datetime
- ✓ datetime-local
- ✓ email
- ✓ month
- ✓ number
- ✓ range
- ✓ search
- ✓ tel
- ✓ time
- ✓ url
- ✓ week

New input attributes in HTML5:

- ✓ autofocus
- ✓ form
- ✓ formaction
- ✓ formenctype
- ✓ formmethod
- ✓ formnovalidate
- ✓ formtarget
- ✓ height and width
- ✓ list
- ✓ min and max
- ✓ multiple
- ✓ pattern (regexp)
- ✓ placeholder
- ✓ required
- ✓ step

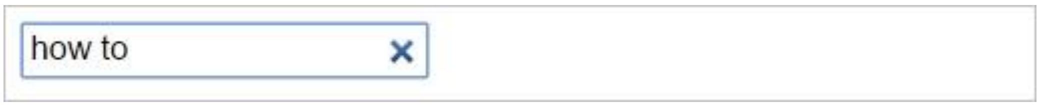
Input types that are not supported by old web browsers, will behave as input type text.

Creating a Search Box

The new search input type can be used to create a search box:

```
<input id="mysearch" name="searchitem" type="search" />
```

Result:



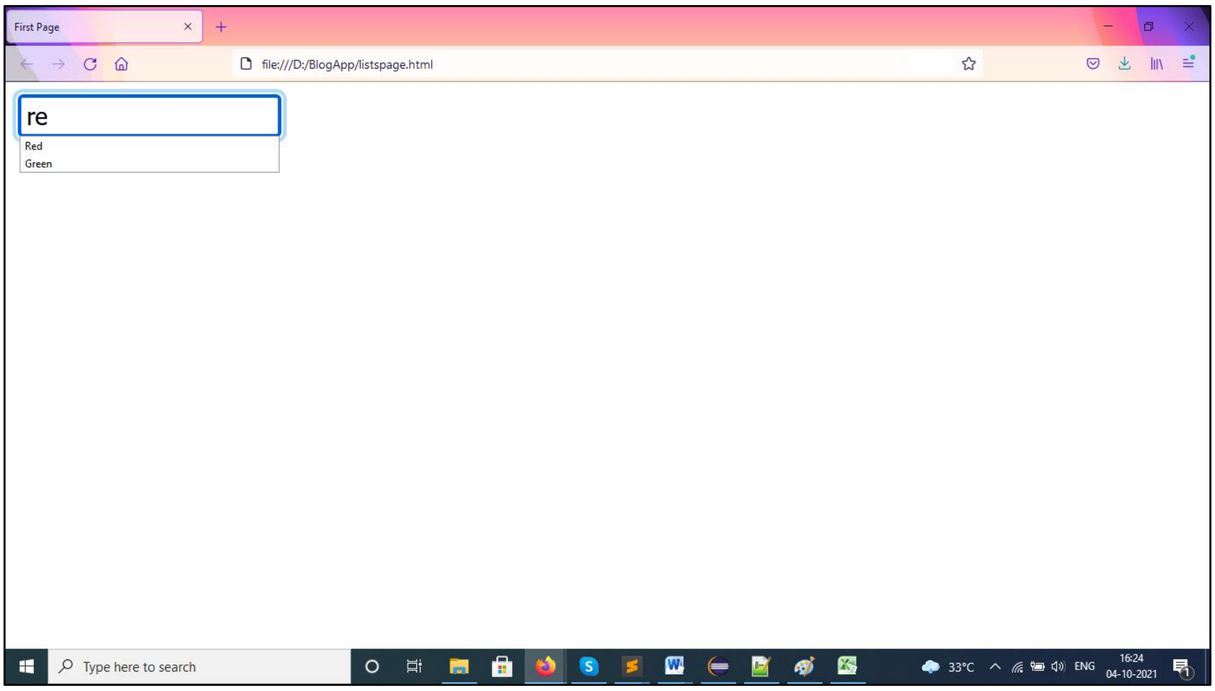
You must remember to set a name for your input; otherwise, nothing will be submitted.

Search Options

The <datalist> tag can be used to define a list of pre-defined options for the search field:

```
<input id="car" type="text" list="colors" />
<datalist id="colors">
<option value="Red">
<option value="Green">
<option value="Yellow">
</datalist>
```

Result:



<option> defines the options in a drop-down list for the user to select.

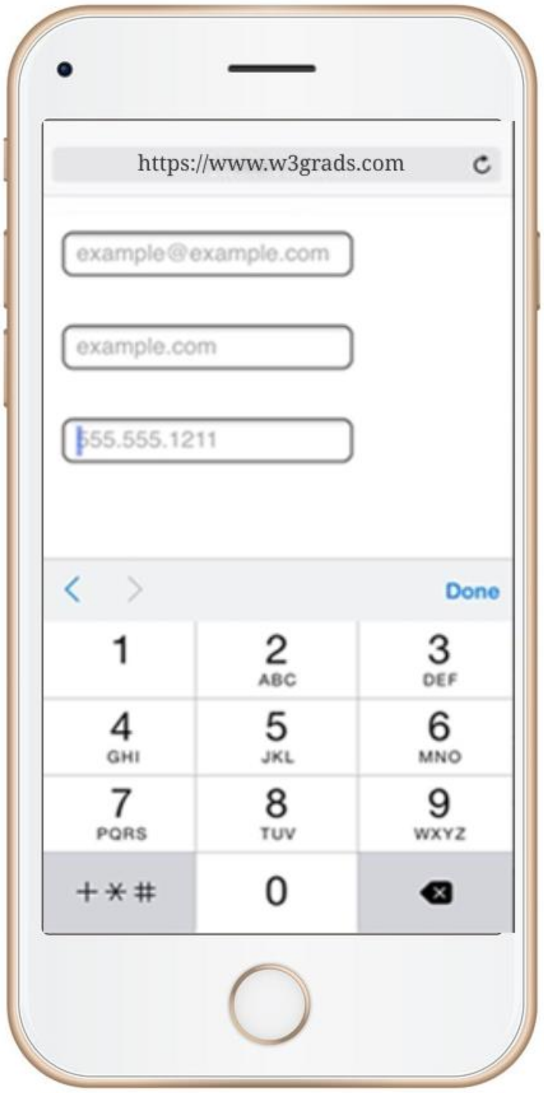
The ID of the datalist element must match with the list attribute of the input box.

Creating More Fields

Some other new input types include email, url, and tel:

```
<input id="email" name="email" type="email"
placeholder="example@example.com" /> <br />
<input id="url" name="url" type="url"
placeholder="example.com" /> <br />
<input id="tel" name="tel" type="tel"
placeholder="555.555.1211" />
```

These are especially useful when opening a page on a modern mobile device, which recognizes the input types and opens a corresponding keyboard matching the field's type:



These new types make it easier to structure and validate HTML forms.