Play with the coding as you play with the football….

WEB-DEVELOPMENT

Mr. adroit
TECHADROIT DELHI

Web-Development

HTML

Html : It stands for Hyper Text Markup Language. HTML is the standard markup language for creating Web pages. HTML describes the structure of a Web page. HTML consists of a series of elements. HTML elements tell the browser how to display the content.

Html Body :-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title></title>
</head>
<body>

</body>
</html>
```

Meta tags: They are used to define the Meta data in an HTML. They are mainly used in SEO (Search Engine Optimization) techniques which help any particular website to rank better in Google or different search engines. It simply boosts the ranking of a webpage to get more traffic on any website.

So, let us understand various meta tags.

```
<meta charset="UTF-8">
```

- It simply means that the characters that are used will be of UTF-8. It declares the page's character encoding. It should contain a standard IANA MIME name for character encodings. Moreover, authors are encouraged to use UTF-8.

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

- It helps any particular website to open in the highest compatibility mode available. It is mostly for those who are still using Internet Explorer. Because there are still some people who have not upgraded their system and are still using the older versions.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- This tag is used to make your website responsive and adjust its width in such a way that it looks good in both PC or mobile. It helps in making the website mobile friendly also.

To add descriptions and keywords on our website, we still use meta tags.

```
<meta name= “description” content = “This is a description”>
```

- To add a description, we have to write the above statement and write your own description in the content part.

```
<meta name= “keywords” content= “html techadroit, graphic design”>
```

Keywords are those special words through which a user reach any website. You can add the keywords in the content part of the tag.

If we want our website to be indexed in Google or other search engines and bots should follow all the links present on the website, then we have to write-

```
<meta name= “robots” content= “INDEX, FOLLOW”>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name= “description” content = “This is a description”>
    <meta name= “keywords” content= “html techadroit, graphic design”>
    <meta name= “robots” content= “INDEX, FOLLOW”>
    <title>Document</title>
</head>
<body>
</body>
</html>
```

To link our stylesheets named as “sanjay.css” in the HTML code, we have to write-

```
<link rel= “stylesheet” href= “sanjay.css”>
```

In the same way, as we have included CSS, we can also include a JavaScript file in <head> tag.

```
<script src = “sanjay.js”></script>
```

Code :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <meta name="description" content="This is description">
    <meta name="keywords" content="html, html development, web development">
    <meta name="robots" content="INDEX, FOLLOW">
    <title>Document</title>

    <!-- This is how you include external css -->
    <link rel="stylesheet" href="sanjay.css">

    <!-- This is how you include external JavaScript -->
    <script src="sanjay.js"></script>
</head>
<body>
</body>
```

```
</html>
```

Image and Anchor Tag

Anchor tag basically helps us to handle links.

In the body tag, if you type **“a"**, **anchor tag** will appear. Just hit the enter key or if you want you can manually write the whole tag. Refer to the declaration below:

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

Here **href** is the attribute of anchor tag where you have to write the URL of the website or Link that you want to open. Next, you have to write the Keyword on which you want the user to click so that he will be redirected to the linked website.

Refer to the example below:

```
<a href="https://google.com">Go to Google</a>
```

Now, the problem arises here is that as soon as we click on the keyword, the website will open on the same webpage but if in case you want to open the website in a new tab then you have to add a new attribute in the anchor tag I.e. **“target".**

You might be confused between tag and attribute, then let me quickly tell you the difference between both.

Tag is like a container that allows you to handle an element whereas **attribute** is the property that enhances that container makes it more convenient to use.

To open a website in a new tab, you have to write something like this:

```
<a href="https://google.com" target = "_blank">Go to Google</a>
```

Through this anchor tag, you can also link internal webpages that are locally available in your directory. You just have to use the address of the internal webpages including the file name with its extension.

For instance:

```
<a href="sanjay.html" target = "_blank">Go to Google</a>
```

So, this was all in the anchor tag of HTML now let’s move to the next topic I.e. handling Images in HTML.

In order to insert an image in a webpage, you have to use the img tag.

An img tag generally have two basic attributes **“src"** and **“alt".**

```
<img src="" alt="">
```

Here **“src"** is the field where you have to insert the URL or address of the image and the “alt" attribute is a field that will display to users if their browser fails to load the image.

In the “alt" tag, we generally input a keyword of the image that can define the image in case the user can’t see the image.

For instance:

```
<img src="https://source.unsplash.com/random" alt= "Error loading image">
```

Here a random image will be loaded because this URL stores multiple images and any one of them gets reloaded every time we refresh the webpage.

Apart from this, let's see another example where we can adjust the height and width of the image using the URL only. Well, this is not something that we can do in every URL. Here we can do it because the developers of the website have created this URL accordingly.

For instance:

```
<img src="https://source.unsplash.com/1600x900/?nature,water" alt= "Error loading image">
```

If you want to manipulate the dimensions of the images, you can use the **“height"** and **“width"** attribute of img tag. Though it is not recommended to manipulate the dimension using these attributes, we will use CSS to alter the design accordingly.

Refer to the illustration below:

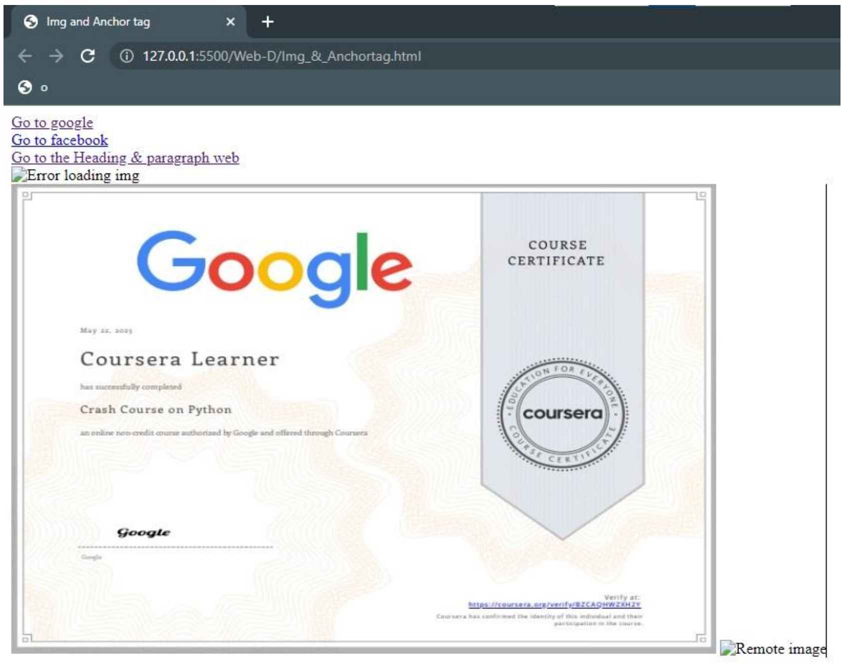
```
<img src="https://source.unsplash.com/1600x900/?nature,water" alt= "Error loading image"
width = "233" height="34">
```

Here the image will be loaded as per the new dimensions.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Img and Anchor tag</title>
</head>
<body>
    <a href="https://google.com">Go to google</a><br>
    <!--To open link in new tab we used 'target' attribute-->
    <a href="https://facebook.com" target="_blank">Go to facebook</a><br>
    <a href="/Web-D/heading_&_Paragraph.html" target="_blank">Go to the Heading &
paragraph web</a><br>
    <!--Image is not present hence alt text is shown-->
    <img src="sanjay.jpg" alt="Error loading img"><br>
    <!--There certificate isamge is present that's why it is not showing alt text-->
    <img src="/Web-D/certificate.jpg" alt="Error loading image" width="720" height="480">
    <!--Unsplash source is a web platform which provides image with free licence-->
    <img src="https://source.unsplash.com/user/wsanter" alt="Remote image" width="720"
height="480">
</body>
</html>
```

Output:



Img and Anchor tag

127.0.0.1:5500/Web-D/Img_&_Anchortag.html

Go to google
Go to facebook
Go to the Heading & paragraph web
Error loading img

Google

May 22, 2023

Coursera Learner

has successfully completed

Crash Course on Python

an online non-credit course authorized by Google and offered through Coursera

COURSE CERTIFICATE

Verify at:
https://coursera.org/verify/BZCAQHWZXH2Y

Coursera has confirmed the identity of this individual and their participation in the course.

Remote image

LISTS and TABLES

The **lists** are basically of two types-

- **Ordered list**

```
<ol>

    <li>This is the first item of my unordered list</li>

</ol>
```

- **Unordered list**

```
<ul>

    <li>This is the first item if my unordered list</li>

</ul>
```

The difference between an ordered and an unordered list is that the **ordered list displays** the list in this format -

1.

2.

3.

….

On the other hand, the **unordered lists display** the list in the following format-

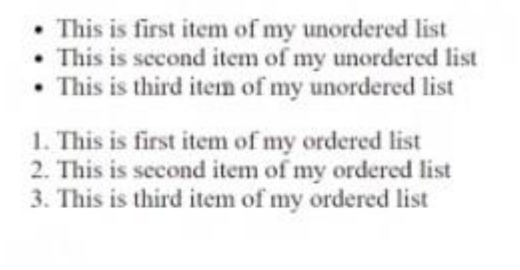
.

.

.

….

Example :



- This is first item of my unordered list
- This is second item of my unordered list
- This is third item of my unordered list

1. This is first item of my ordered list
2. This is second item of my ordered list
3. This is third item of my ordered list

Both the lists have more than one attribute which we can write using the type command. For example, if we write:

```
<ol type= “I”>
```

Then we will get the lists as I, II, III, and so on. In the same format, we can also get the lists as A, B, C, and so on.

This applies on unordered lists also. If we write

```
<ul type= “square”>
```

Then we will get a bulleted square instead of a circle. There are various other references available from which you can see all the attributes.

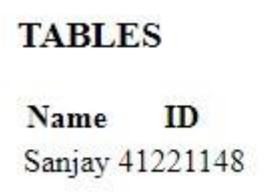
HTML also allows the nesting of lists. It simply means we can add a list into another list.

```
<ul type="circle">
    <li>This is first item of my unordered list</li>
    <li>This is second item of my unordered list</li>

    <ul>
        <li>Another one</li>
        <li>Another two</li>
        <li>Another three</li>
</ul>
```

Let us now discuss the **Tables in HTML**. A table is just a combination of rows and columns on a webpage. The structure of the table looks like-

The main part is the table tag, and it consists of two parts: the table head and table body. The <thead> consists of the main head of the table and <tbody> consists of the body of the table.

<tr> is used to justify that it is the part of a row. Inside the <tr> tag, we give the headings of a row under the <th> tag. The final structure of a table looks like-



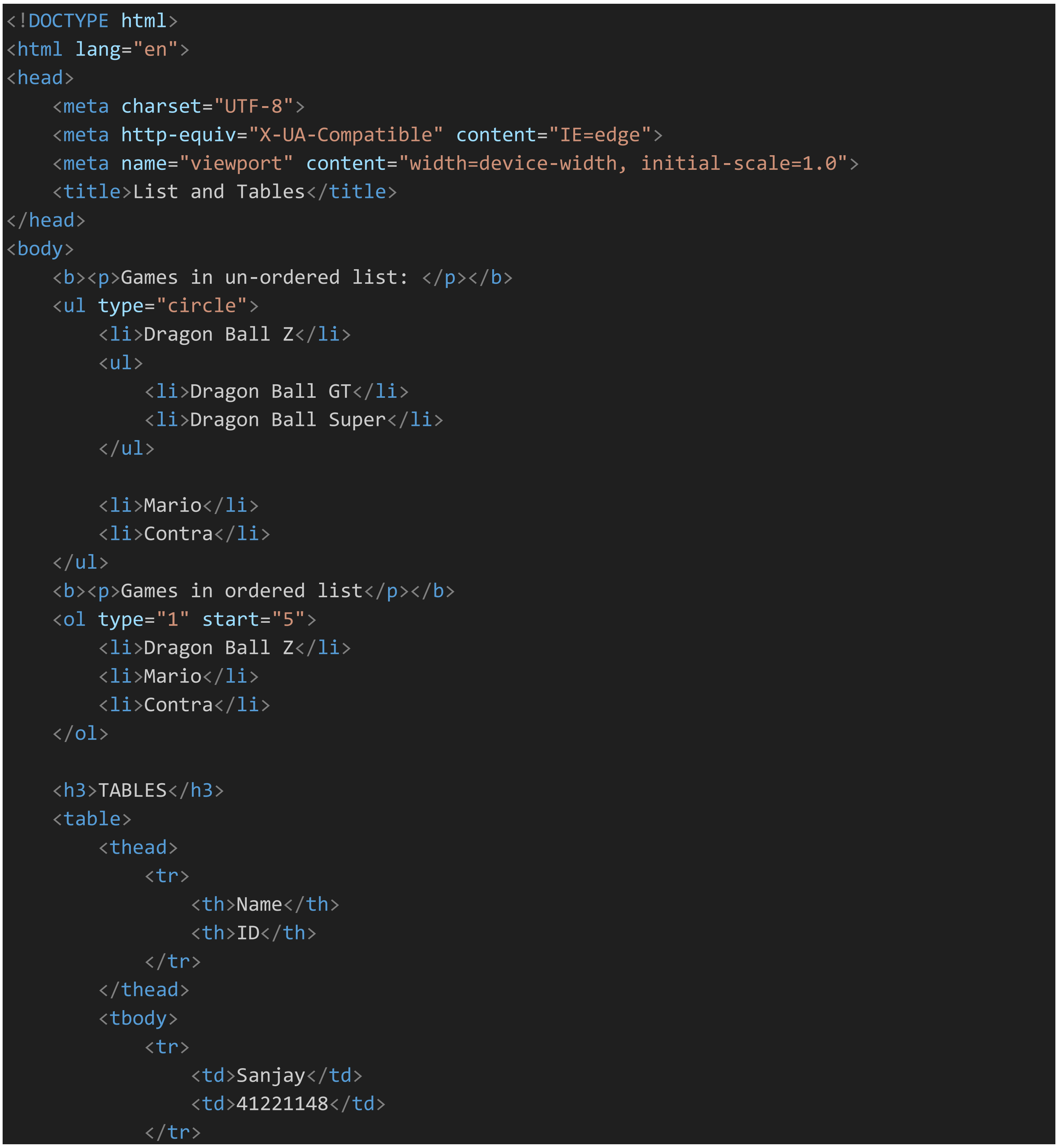
TABLES

Name	ID
Sanjay	41221148

In a table, there are mostly two things to remember, the head and the body of a table. To add more rows to the table, we can simply add a <tr> tag and add any number of rows in a table.

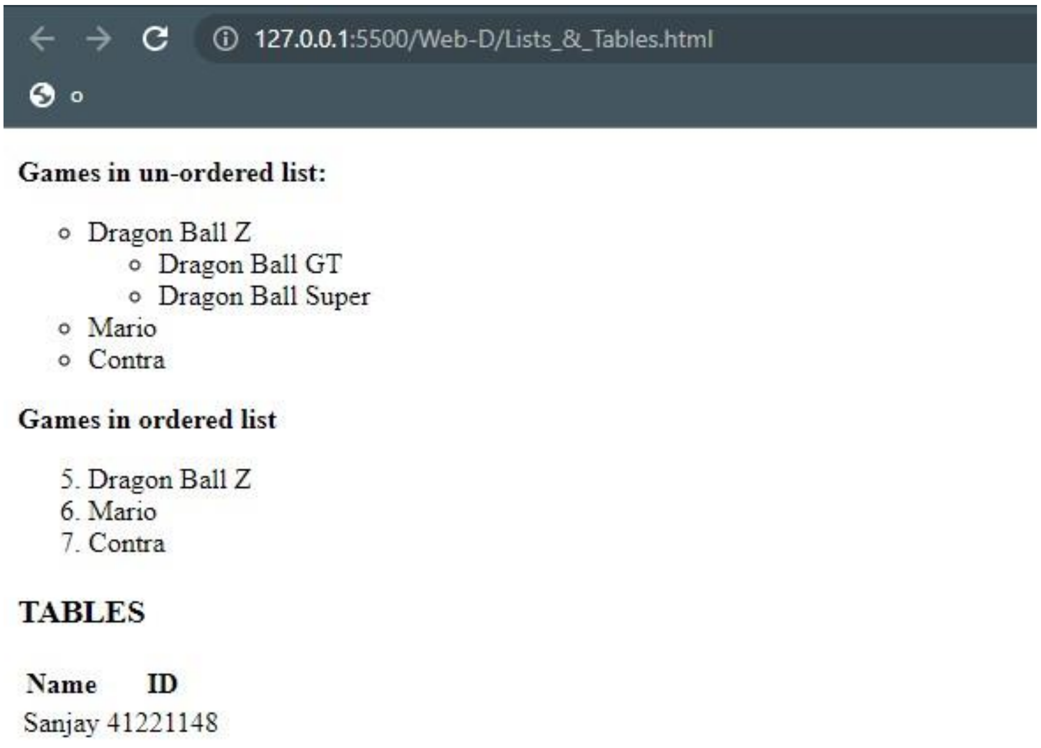
Tables and lists are two primary components of a website that helps in making it more attractive.

CODE:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>List and Tables</title>
</head>
<body>
    <b><p>Games in un-ordered list: </p></b>
    <ul type="circle">
        <li>Dragon Ball Z</li>
        <ul>
            <li>Dragon Ball GT</li>
            <li>Dragon Ball Super</li>
        </ul>

        <li>Mario</li>
        <li>Contra</li>
    </ul>
    <b><p>Games in ordered list</p></b>
    <ol type="1" start="5">
        <li>Dragon Ball Z</li>
        <li>Mario</li>
        <li>Contra</li>
    </ol>

    <h3>TABLES</h3>
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>ID</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Sanjay</td>
                <td>41221148</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

Output:



127.0.0.1:5500/Web-D/Lists_&_Tables.html

Games in un-ordered list:

- Dragon Ball Z
  - Dragon Ball GT
  - Dragon Ball Super
- Mario
- Contra

Games in ordered list

5. Dragon Ball Z
6. Mario
7. Contra

TABLES

Name	ID
Sanjay	41221148

<u>FORMS AND INPUT TAGS</u>

Forms are a very important part of HTML. It represents the document section containing interactive controls for submitting information.

Whenever we add a **<form>** tag in the HTML, it is going to ask for some action for submitting that particular form in the backend for future reference. So, for now, we will write it as **backend.php**. All the data submitted in a form will be stored automatically in the backend "backend.php" after submitting it.

The template will look like-

```
<form action= "backend.php">
```

Then comes the **<input>tags** which are present inside the form, where the user provides the input. These inputs can be of any type whether text, button, checkbox, date, time, etc. Input Tags are used to create interactive controls for web-based forms in order to accept data from the user.

The **<span>** is an in-line element and <div> is a block element. Which means, if we use two separate div tags for different inputs, then all the inputs will come on different lines.

To get the input as text, the syntax is-

```
<input type= “text”>
```

To get the input type as an email in the form, the syntax is-

```
<div>
    Email: <input type="email" name="myEmail">
</div>
```

The name here is used so that the backend can recognize the tag that we are using.

To get the submit button in the form, the syntax is-

```
<div>
    <input type= “submit” value= “submit now”>
</div>
```

We can also add date and time in the form. To add these, the general syntax is-

```
<div>
    <input type= “date” name= ”myDate” id= “”>
</div>
```

It will give the complete date form in the format of **“dd/mm/yyyy”.**

To add any numeric text in the HTML form, the syntax is-

```
<div>
    Number: <input type= “number” name “myNumber”>
</div>
```

While filling several online forms, you must have seen the **radio buttons** and **checkboxes** in the form. Radio buttons are such buttons that allows to select any one of the following options amongst all. For example, while selecting the gender, we can only select either male or female. Whereas the checkbox allows selecting the multiple options available. The example of both the formats are as follows-

- **For checkbox-**

```
<div>
    Are you eligible?: <input type="checkbox" name="myEligibility" checked>
</div>
```

- **For Radio buttons-**

```
<div>
    Gender: Male <input type="radio" name="myGender"> Female <input type="radio"
name="myGender">
    Other <input type="radio" name="myGender">
</div>
```

- To reset all the information, entered in the form, we take the help of a reset button. To get the reset button, we have to write-

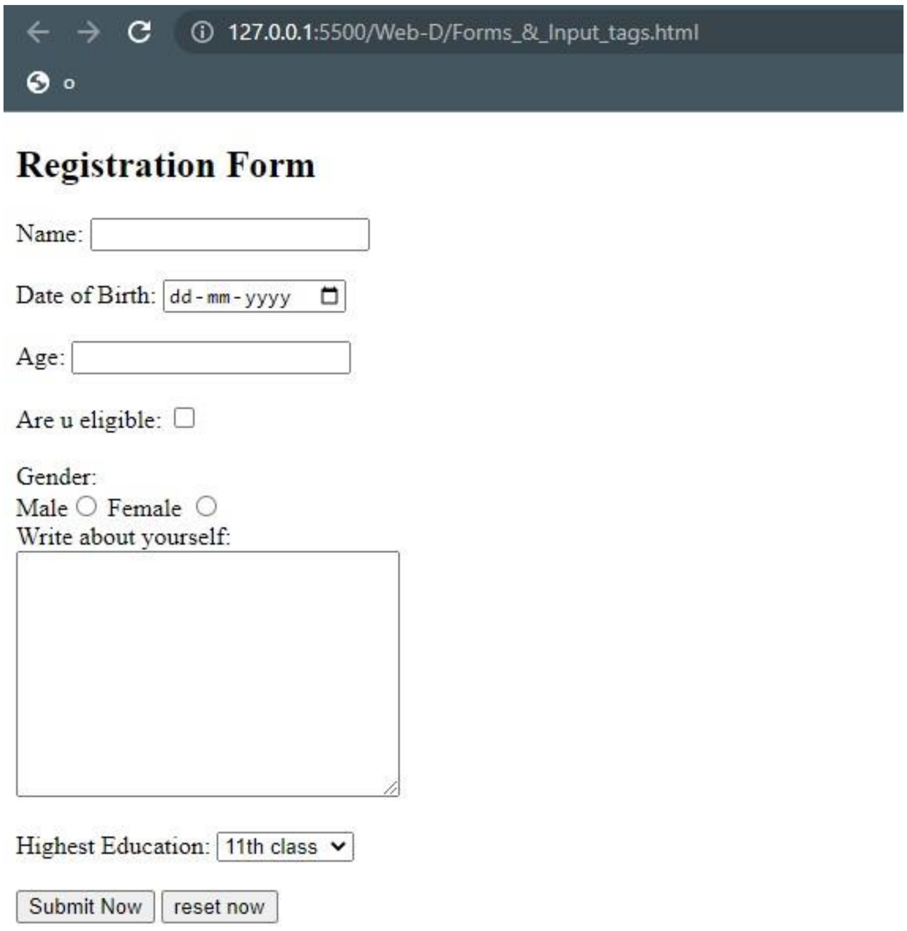
```
<div>
    Input type= “reset” value= “Reset Now”
</div>
```

CODE:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Forms and Input Tags</title>
</head>
<body>
    <form action="backend.php">
    <h2>Registration Form</h2>
    <div> <label for="Name">Name: </label>
          <input type="text">
    </div>
    <br>
    <div>
       Date of Birth: <input type="date" name="date">
    </div>
    <br>
    <div>
       Age: <input type="number" name="age" id="">
    </div>
    <br>
    <div>
        Are u eligible: <input type="checkbox" name="eligibility" id="">
    </div>
    <br>
    <div>
        <!--Give the same name if u want to choose any one-->
        Gender: <br> Male<input type="radio" name="mygender" id=""> Female <input
type="radio" name="mygender" id="">
    </div>
    <div>
        Write about yourself: <br><textarea name="about" id="" cols="30"
rows="10"></textarea>
    </div>
    <br>
    <div>
        <label for="Education">Highest Education: </label>
        <select name="education" id=""><option value="11th">11th class</option>
        <option value="12th">12th class</option>
        <option value="graduation">graduation</option>
        </select>
    </div>
    <br>
    <div>
        <input type="submit" value="Submit Now" >
        <input type="reset" value="reset now">
    </div>

</form>
</body>
```

```
</html>
```

Output:


127.0.0.1:5500/Web-D/Forms_&_Input_tags.html

Registration Form

Name:

Date of Birth: dd-mm-yyyy

Age:

Are u eligible:

Gender:
Male Female
Write about yourself:

Highest Education: 11th class

INLINE AND BLOCK ELEMENTS

Inline elements are those elements which only occupy the space bounded by the tags defining the element, instead of breaking the flow of element. On the other hand, block-level elements take up the entire space of its parent element. Let us understand this with an example-

If we write any text in the paragraph tag like this-

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

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

OR

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

In both the above examples, we will see the output in both different lines, not in the same line. We want both the texts in the same line but it is not so. Can you think why?

It is because the paragraph tag is a block element. The Block element means that it will take the full width of a single line and does not allow any other content to fit in it. But, if we write both the texts between the **<span> tags** like-

```
<span>This is a paragraph</span> <span>This is also a paragraph</span>
```

Then we see that both the texts will appear in the same line. It is because the <span> tag is an inline element. It allows all the elements in the same line.

To understand it more, we can take the help of CSS by applying the border. However, you need not worry about the border as it is a part of CSS.

```
<p style= “border: 2px solid, red;”>This is a paragraph</p> <p style= “border: 2ox solid blue;”>This is also a paragraph</p>

<span style= “border: 2px solid red;”>This is a span</span> <span style = “border: 2px solid red;”>This is also a span</span>
```

After testing the above code in the live server, you will know the main difference between inline and block elements.

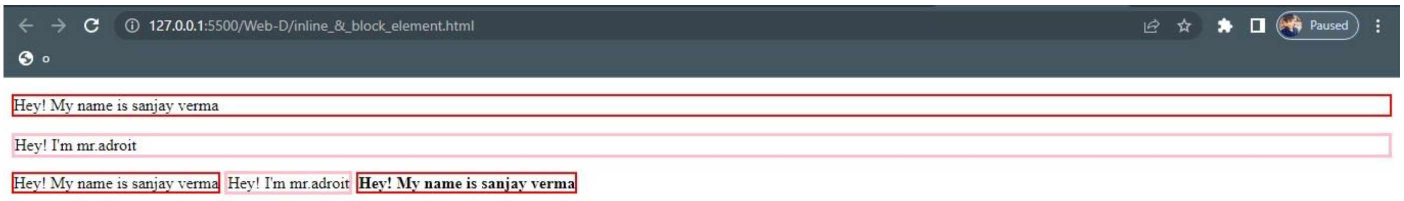
The above example shows how inline elements will take only its portion of text whereas the block element will take the whole width of the line. You will understand it more with the help of different colours through CSS. Anchor <a> tags also behaves like an inline element.

We have two different options of making our text appear in a single line. The first one is either with the help of CSS through borders or with the help of inline elements.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inline and block element</title>
</head>
<body>
    <p style="border:2px solid red;">Hey! My name is sanjay verma</p><p style="border: 3px solid pink;">Hey! I'm mr.adroit</p>
    <span style="border: 2px solid red;">Hey! My name is sanjay verma</span> <span style="border: 3px solid pink;">Hey! I'm mr.adroit</span>
    <strong style="border:2px solid red;">Hey! My name is sanjay verma</strong>

</body>
</html>
```

Output:



```
127.0.0.1:5500/Web-D/inline_&_block_element.html

Hey! My name is sanjay verma

Hey! I'm mr.adroit

Hey! My name is sanjay verma  Hey! I'm mr.adroit  Hey! My name is sanjay verma
```

IDS & CLASSES

When a new child is born, we urge to give him a name or his identity by which he will be known further. Or if you are having a pet, you must have given him some name to call. In the same way, **IDs** refer to giving a name to any particular element for its identity. It simply refers giving an identity to an element. We know, no two names can be given to any of the two members of the family. In the same way, one ID can be given to only one element on a website. Therefore, in the below example, the id **mainBox** cannot be given to any other element.

```
<div id= “mainBox” class= “redBG”>
```

Now the question arises what is the need for an ID in HTML? The answer is, while using JavaScript or CSS, we can target one full element and can make the necessary changes in it. In the same way, we can grab the full element and change the border or width or many more things through CSS.

Let us now understand what are classes with an example. Assume that I am having 100 elements in my HTML and I want to give a red background to all the 100 elements. To do this, we have two options. Either we have to select each element and assign a red background to it or we can create a class redBG and assign a red background to it. Then we can give this class to the elements in which we want a red background color. To avoid confusion, I am assuming that the class redBG is already defined.

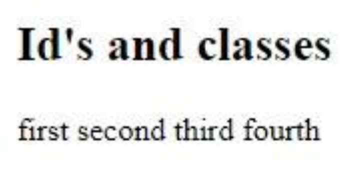
One point to note here is we can assign only one ID to a particular element but it is not so in the case of classes. An element can have more than one class in itself. The more classes we add in an element, the more property will get added to it.

Classes are denoted by a **dot ‘.’ and ID is denoted by hash** ‘#’. For example, to get a redBG class in an element we can simply write that element name followed by .redBG. The below picture shows everything, you have learned till now-

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ID's and Classes</title>
</head>
<body>
    <h2>Id's and classes</h2>
    <div id="mailbox" class="redbg">
    </div>
    <!-- . for class-->
    <!-- # for id-->
    <div class="redbg blackborder pinkline">
    </div>
    <!-- span.myClass.myClass2.myClass3*4 + <tab> -->
        <span class="myClass myClass2 myClass">first</span>
        <span class="myClass myClass2 myClass">second</span>
        <span class="myClass myClass2 myClass">third</span>
        <span class="myClass myClass2 myClass">fourth</span>
</body>
</html>
```

Output:


Id's and classes

first second third fourth

HTML ENTITIES

You will normally say that all the extra spaces will reflect back on the webpage. But it is not so. Because HTML treats all the extra spaces as a single space only and automatically removes all of them. Therefore, if you want to use extra spaces or any special characters, then you have to use HTML entities. To get extra space we can use &nbsp (non-breaking space) after that particular text. For example-

```
<div class= “container”>
    <p>This is another &nbsp Paragraph</p>
</div>
```

By writing this way, we can create 1 extra space after the word another. By adding five &nbsp, we can get 5 extra spaces.

Entities are also used to write some special characters that you cannot write from keyboards and also those words that are reserved in HTML. For example, if we want <p> to appear in the result, then it is not possible without the help of entities.

```
<div class= “container”>
    <p>This is another Paragraph<p></p>
</div>
```

By writing the above code, we will see that we cannot obtain the <p> in our result. Therefore, the solution for this is we can take the help of HTML entities. We have to write in this format-

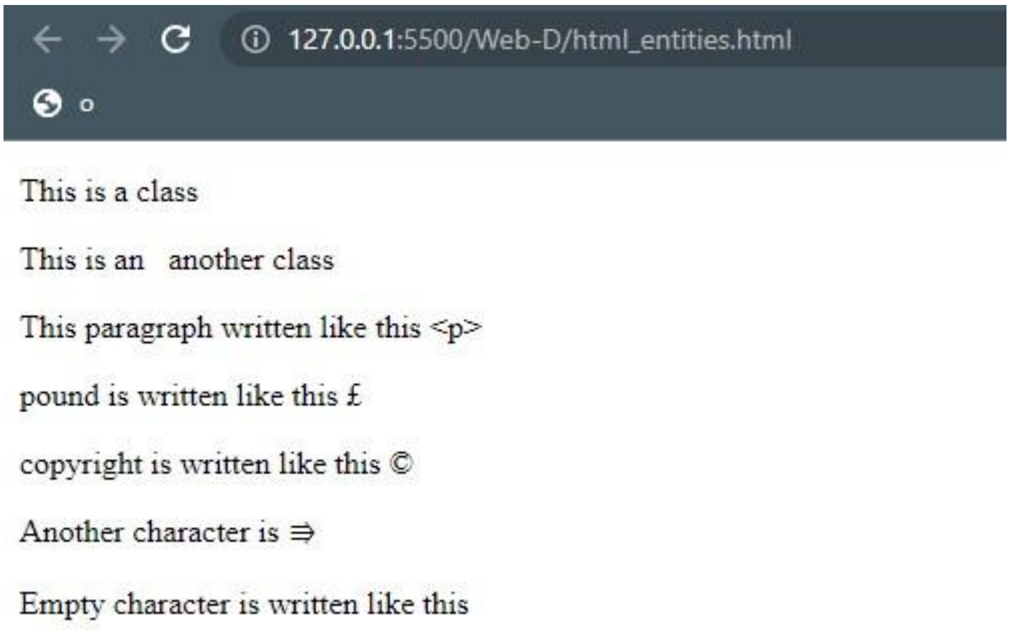
```
<div class= “container”>
    <p>This is another Paragraph &lt;p&gt; </p>
</div>
```

There is a list of different reserved characters and hundreds of special characters that you cannot write without entities.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML entities</title>
</head>
<body>
    <div class="container"><p>This is a class</p></div>
    <div class="container"><p>This is an &nbsp; another class</p>
    <p>This paragraph written like this &lt;p&gt;</p>
    <p>pound is written like this &pound;</p>
    <p>copyright is written like this &copy;</p>
    <p>Another character is &rAarr;</p>
    <p>Empty character is written like this &#8203;</p>
</div>
</body>
</html>
```

Output:



127.0.0.1:5500/Web-D/html_entities.html

```
This is a class

This is an   another class

This paragraph written like this <p>

pound is written like this £

copyright is written like this ©

Another character is ⇛

Empty character is written like this
```

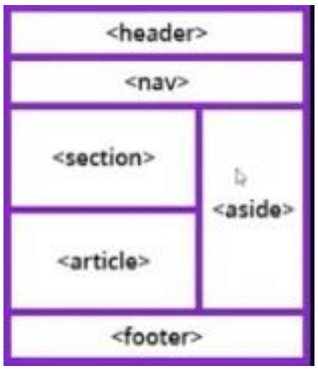
SEMANTIC TAGS

there are different tags used for the headers, navigation bars, body, and footers. So to analyze each and every tag properly, we have to use Semantics. The better the use of Semantics in a website, the more is the number of chances that it is being crawled by the search engines. It will also help in the better ranking of a website.

In very simple language, Semantic means to provide meaning to any word. If we talk in about a web page, then earlier most of the websites are made only using divs including different classes and IDs in them. But that was not the correct way of telling the search engines that if the elements are header, footer, or any other body.

There are many examples of Semantics. You can take the reference of the internet to view all. Here are some of the important semantic elements-

- **<header>**
- **<nav>**
- **<section>**
- **<article>**
- **<footer>**



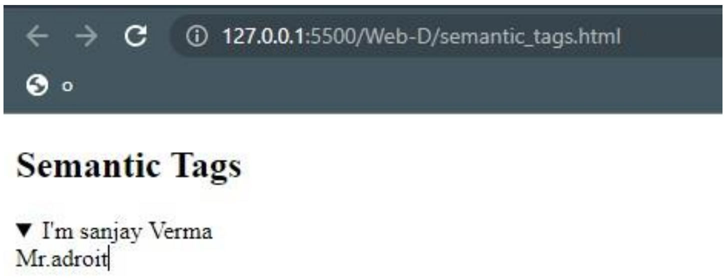
With these examples, the search engines can easily differentiate between headers, navigation bars, sections, bodies, and footers. Also, when we share the link of this website in social media, then the viewers will easily understand headings, sections, and body of a website.

But now the question arises, is it necessary to use all these things? The straight answer is No. It is not necessary to use the semantics but including it will help in the SEO part and increase the probability of ranking of your website. If you are working on a blog, then you must use this technique to improve its ranking. On the other hand, non-semantic elements do not justify their meaning. For example, div and span don't tell much what they do apart from the division between texts.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Semantic tags</title>
</head>
<body>
    <h2>Semantic Tags</h2>
    <details>
        <summary>I'm sanjay Verma</summary>
        Mr.adroit
    </details>
</body>
</html>
```

Output:


127.0.0.1:5500/Web-D/semantic_tags.html

```
Semantic Tags

▼ I'm sanjay Verma
Mr.adroit
```

INTRODUCTION TO CSS

CSS stands for Cascading Style Sheets and is used to design the website to make it look attractive.

Let us first understand, what is CSS?

- CSS gives style to raw HTML
- It stands for Cascading Style Sheets
- CSS is used to give style to our web pages
- CSS is used to make websites responsive
- CSS takes the responsibility of design in your websites

CSS includes all the things which can be used to design the raw HTML from colouring the background and texts, to adjust the borders, give padding, etc. Moreover, CSS helps in making websites responsive. Responsive means that the site will behave accordingly to the different screen sizes. For example, if you open a website on a desktop and then on your mobile, you will find the difference between their displays. All the components in a navigation bar will move into a hamburger icon if you open the website on mobile.

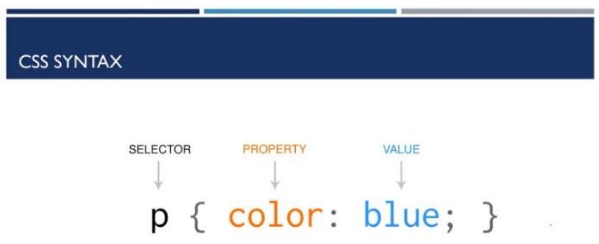
We can add styles in the HTML part itself, but I would rather recommend making a new CSS file and then attach it to the HTML part. It is so because it is a professional practice when different developers are working on a single website to keep the skeleton of a website in one file and the styling in another file.

Role of CSS

- CSS is a style sheet language that is used to handle the presentation of the web page containing HTML.
- It makes our websites beautiful and modern looking.

CSS Syntax

The syntax of CSS is-

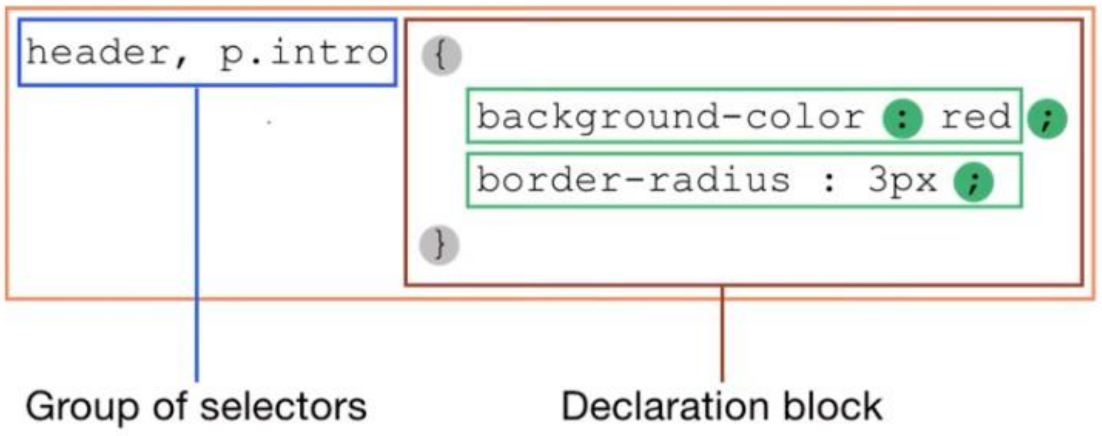


- P stands for the selector and it decides which part of the HTML the CSS will be applied. It states where the CSS property is to be applied.
- Property is used to describe which property you want to change or add. Whether you have to change colour, border, background, width, all these come under property.
- The last section is for defining the value. All the properties will be changed according to the value we provide.

We can also target multiple properties at one time. The syntax is as follows-

```
header, p.intro { background-color: red;
    border-radius: 3px,
}
```

CSS SYNTAX



In the above example, we have changed the header tag and the paragraph tag with a class intro to change the background colour to red and border-radius to 3 pixels.

There are three ways to Add CSS-

1. Inline CSS- CSS is added to the elements directly using the style attributes.
2. Internal CSS- CSS is kept inside the head tags in <style> tags
3. External CSS- CSS is kept separately inside a .CSS style sheet. It involves two steps-

- First, write the CSS in.CSS file.
- Include that CSS file to Markup.

INLINE, INTERNAL & EXTERNAL CSS

- **Inline CSS** allows you to apply a unique style to one HTML element at a time. You can assign the Inline CSS to a specific HTML element by using the style attribute with any CSS properties defined within it. Let us try to understand this with an example.

```
<body>
    <h1>This is CSS </h1>
     <p style= “color: red;”>This will teach you everything you need to know about CSS</p>
</body>
```

You must be thinking this is the best way to add CSS on the website but I will let you know that it is not the best method to style your HTML. If you add too much Inline CSS, then your HTML will become too messy to understand for you.

- **Internal CSS** is used to define a style tag for a single HTML page. It is defined in the <head> section within a <style> element. Let us understand the External CSS with the help of an example.

```
<head>
    <style>
             p{
                  color: purple;
                 }
    <style>
<head>
```

One important point to note here is, Inline CSS is given more priority than Internal CSS.

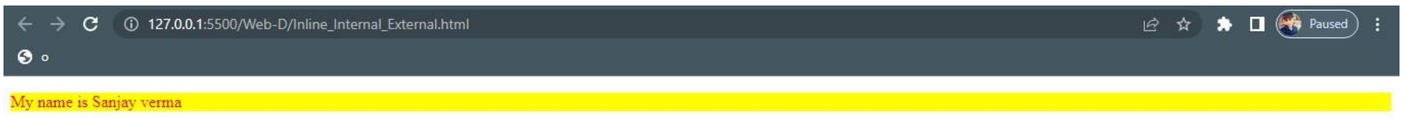
- **External CSS** is mostly used when you want to make changes on multiple pages. It is an ideal condition because it facilitates you to change the look of the entire website by changing just one file. We will add the stylesheet in the <head> section using <link> tag.

```
<head>
    <link rel= “stylesheet” href= “tech.css”>
<head>
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inline, Internal and External CSS</title>
    <!--Internal CSS-->
    <style>
        p{
            color: red;
        }
    </style>
    <!--External CSS-->
    <link rel="stylesheet" href="External.css">
</head>
<body>
    <!--Inline CSS-->
   <!-- <p style="color: purple; background-color: orange; font: large;">I'm Mr.adroit</p>
-->
    <p >My name is Sanjay verma</p>
</body>
</html>
```

Output:



My name is Sanjay verma

SELECTORS

CSS selectors are used to select any content you want to style. These are the part of CSS ruleset. CSS selectors select HTML elements according to its id, class, type, attribute, etc.

- CSS Selectors are used to target HTML elements.
- Selectors make it easy for us to easily target single/multiple HTML elements in the markup.

We will see four types of CSS Selectors:

- CSS element Selector

- CSS id Selector
- CSS class Selector
- The CSS grouping Selector

As discussed in one of the previous videos, the basic syntax of writing the CSS is-

p {color: blue;}

In the example above, ‘p’ is the selector. It will convert all the paragraph into blue.

So let us now start by making a new file as tut14.html and as usual, add an instant boilerplate Visual Studio Code. Give the title as CSS Selectors in the <title> tag. In this example, we will be using Internal CSS, not Inline CSS. However, you can also use External CSS. I will be explaining using internal CSS as I want everything to be within the page. Let us start with the simple example-

- **Element Selector**

```
<h3>CSS Selectors</h3>
             <p id="firstPara">This is a simple paragraph to demonstrate css selectors</p>
             <p id="secondPara" class="redElement bgBlue">This is a another simple
paragraph to demonstrate css selectors</p>
<div>
       <p>This is yet another simple paragraph inside div to demonstrate css selectors</p>
</div>
```

- **Class Selector-** If we want to select a paragraph and assign multiple properties to it, then we can use Class Selector. Let us understand with an example-

```
<style>
    .redElement{
            Color: red;
                }
.bgBlue{

     Background-color: blue;
}
</style>
```

```
<body>
    <h3>CSS Selectors</h3>
    <p>This is a simple paragraph to demonstrate css selectors</p>
    <p id="secondPara" class="redElement bgBlue">This is a another simple paragraph to
demonstrate css selectors</p>
    <div>
        <p>This is yet another simple paragraph inside div to demonstrate css
selectors</p>
    </div>
</body>
```

- **ID Selector-** If we want to select the only paragraph to show any change, then we will be using ID selector. Let us understand with an example-

```
<style>
    #firstPara{
             color: green;
                }
</style>
```

```
<body>
    <h3>CSS Selectors</h3>
    <p id="firstPara">This is a simple paragraph to demonstrate css selectors</p>
    <p>This is a another simple paragraph to demonstrate css selectors</p>
    <div>
        <p>This is yet another simple paragraph inside div to demonstrate css 
selectors</p>
    </div>
</body>
```

- **Grouping Selector-** Grouping Selector is used when we have to make changes in more than one element. Let us understand with an example. Suppose we have two elements footer and span and we want the same changes in both the elements. Then we can do the following-

```
<style>
    footer, span{
        Background-color: pink;
                }
</style>
```

```
<body>
    <h3>CSS Selectors</h3>
    <p>This is a simple paragraph to demonstrate css selectors</p>
    <p>This is a another simple paragraph to demonstrate css selectors</p>
    <div>
        <p>This is yet another simple paragraph inside div to demonstrate css 
selectors</p>
        <span>this is span</span>
    </div>
    <footer>This is footer</footer>
</body>
```

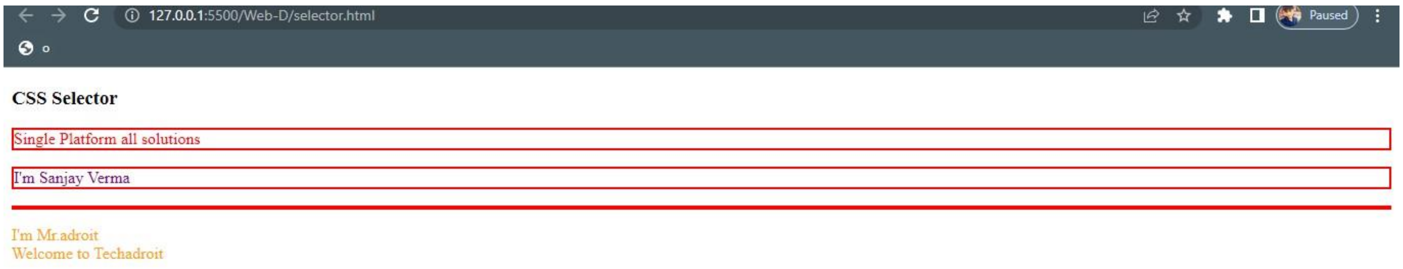
So, I believe, you must have understood the basic concepts of CSS Selectors. Till now, you must keep two points in your mind-

- There are three ways of writing CSS- Inline, Internal, and External.
- How to do the basic selections of CSS selectors.

CODE:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS selector</title>
    <style>
        /* Element selector */
        p{
            color: red;
            border: 2px solid red;
        }
        /* Id selector */
        #secondpara{
            color: purple;
        }
        /* class selector */
        .Third{
            color: blue;
        }
        /* class grouping selector */
        span, footer{
            color: orange;
        }
    </style>
</head>
<body>
    <h3>CSS Selector</h3>
    <p>Single Platform all solutions</p>
    <p id="secondpara">I'm Sanjay Verma</p>
    <div>
        <p id="thirdpara" class="Third"></p>
        <span>I'm Mr.adroit</span>
    </div>
    <footer>Welcome to Techadroit</footer>
</body>
</html>
```

Output:



CSS Selector

Single Platform all solutions

I'm Sanjay Verma

I'm Mr.adroit
Welcome to Techadroit

FONTS IN CSS

```
<body>
    <h4>CSS Fonts</h4>
   <p>Lets play with <span>fonts</span>. It is very exciting</p>
</body>
```

This is a very basic code as an example to start playing around different fonts. In CSS, we have **two** types of **fonts- web-safe fonts** and **web fonts**. Web saved fonts are the fonts that come pre-installed with most of the operating systems, therefore, using these fonts you will never encounter any error. But on the other hand, some fonts are not shipped with the OS; so to use them, we need to import them from the web.

We can also use the technique of font stack. A font stack is a list of fonts that are listed in order of preference you would like them to appear in case some fonts are not loading. The example of this is shown below-

```
p {font-family:'Ubuntu', 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;}
```

This list will be iterated until the specified font is not available in the system.

To see the whole list of web saved fonts, there is a very good website called CSS Font Stack. It provides the complete list of web saved fonts. Talking about web fonts, we can easily import them from Google. To import the code, there is no website better than Google Fonts. To use it, simply copy the style-sheet and add it to your code and update the font stack with the specific font you desire.

- The next property is font size.

```
p { font-family: ‘Franklin Gothic Medium’, ‘Aerial Narrow’, Aerial, sans-serif;
    font-size: 33px;
}
```

Font Size is used to set the size of a font. In the above example, we used our font size to be 33px. Pixel ‘px’ is the unit of the font size and it is 1/96th of an inch.

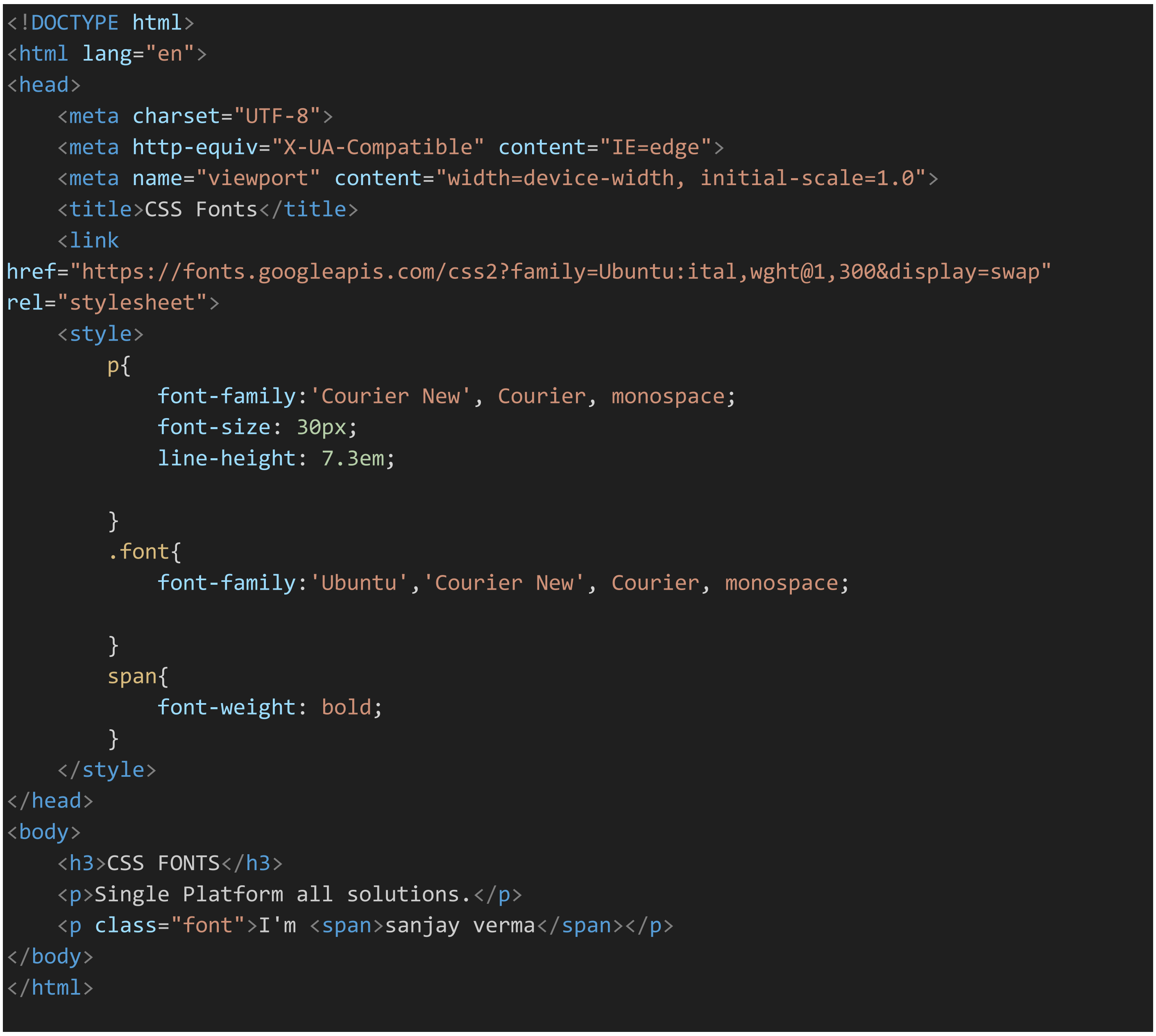
- The next property is line-height. Line-height is the spacing between the fonts (current font and previous font).

```
p {font-family: ‘Franklin Gothic Medium’, ‘Aerial Narrow’, Aerial, sans-serif;
    font-size: 23px;
    line-height: 1.8em;
}
```

- Next property is font-weight. The font weight property sets how thick or thin character in text should be displayed.

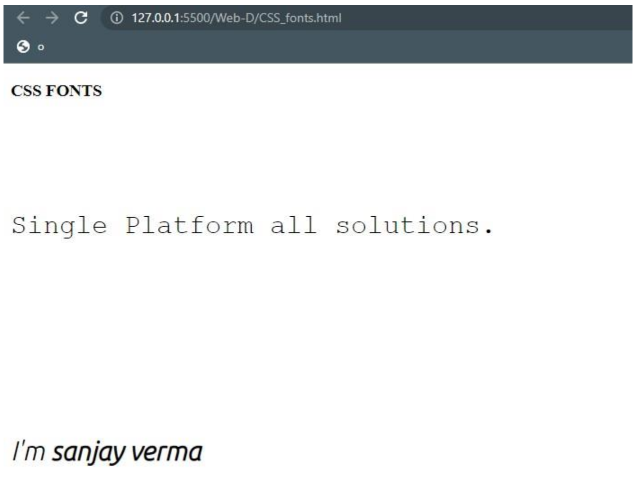
```
p{ font-family: ‘Franklin Gothic Medium’, ‘Aerial Narrow’, Aerial, sans-serif;
    font-size: 23px;
    line-height: 1.8em;
    font-weight: bold;
}
```

CODE:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Fonts</title>
    <link
href="https://fonts.googleapis.com/css2?family=Ubuntu:ital,wght@1,300&display=swap"
rel="stylesheet">
    <style>
        p{
            font-family:'Courier New', Courier, monospace;
            font-size: 30px;
            line-height: 7.3em;

        }
        .font{
            font-family:'Ubuntu','Courier New', Courier, monospace;

        }
        span{
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h3>CSS FONTS</h3>
    <p>Single Platform all solutions.</p>
    <p class="font">I'm <span>sanjay verma</span></p>
</body>
</html>
```

Output:



```
127.0.0.1:5500/Web-D/CSS_fonts.html

CSS FONTS

Single Platform all solutions.

I'm sanjay verma
```

COLORS IN CSS

```html
<body>
    <h2>This is my first box</h2>
    <p id="firstPara">This is a paragraph from first box</p>

    <h2>This is my first box</h2>
    <p id="secondPara">This is a paragraph from second box</p>

    <h2>This is my first box</h2>
    <p id="thirdPara">This is a paragraph from third box</p>
</body>
```

1. The first method of defining the colour in the CSS is directly writing the particular colour name. Its example is

```css
#firstPara{
    color:blueviolet; /* Color by name */
}
```

Using the above code, we have changed the colour of our paragraph to blue-violet.

2. The second way of defining colour is with the help of 'RGB,' as shown below. The RGB colour range varies from 0 to 255.

Here, we treat RGB as a function and pass the values in that function for red, green and blue. As we give different values in the function, it creates various combinations of different colours combined with red, green, and blue.

3. The third way is by giving hex colours. However, it is very rarely used.

```css
#thirdPara{
    color: white;
    background-color: #ff4532; /* Color by hex value */
}
```

In the above code, we can see the **"#"** character. It is used to give the hexadecimal value of any colour. You will find various references on the Internet to generate the hexadecimal values of different colours.

Let us now understand the working of hex colours. Hex colour is also a kind of RGB. For example, let's take one hex value as **'#60DCA4'**, here 60 is red of RGB, DC is green of RGB, and A4 is the blue of RGB. The same value in RGB for this colour would be something **(96, 220,164).**

Colour picker is one of the most interesting things that you will find in VS code by using any of the colour types. You can select any of the colours from the colour picker, and the values will automatically get set by the colour picker for that particular colour type.

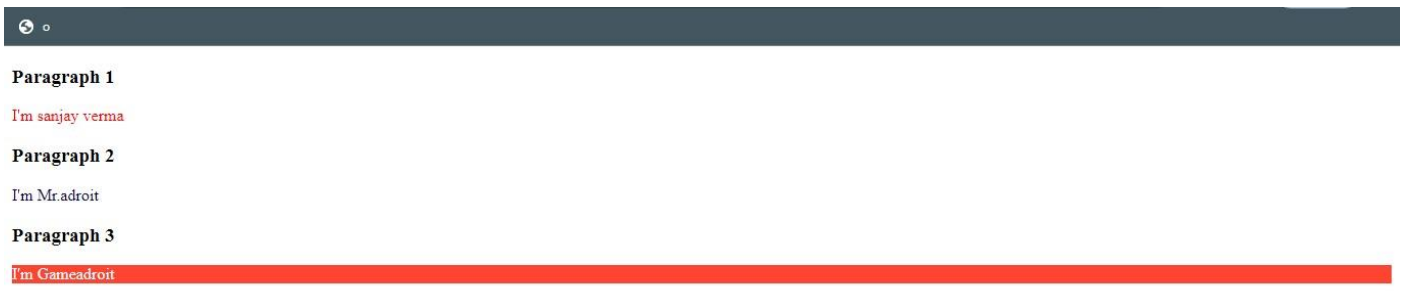
The background colour also works the same way as the text colour works, for example:

```css
#secondPara{
    background-color: rgb(0,0,0);
 }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Colors in CSS</title>
    <style>
        #para1{
            color: red;
        }
        #para2{
            color: rgb(8, 9, 71)
        }
        /* Hex color */
        #para3{
            color: #fff;
            background-color: #ff4532;
        }
    </style>
</head>
<body>
    <h3>Paragraph 1</h3>
    <p id="para1">I'm sanjay verma</p>
    <h3>Paragraph 2</h3>
    <p id="para2">I'm Mr.adroit</p>
    <h3>Paragraph 3</h3>
    <p id="para3">I'm Gameadroit</p>
</body>
</html>
```

Output:



Paragraph 1

I'm sanjay verma

Paragraph 2

I'm Mr.adroit

Paragraph 3

I'm Gameadroit

BORDERS & BACKGROUNDS

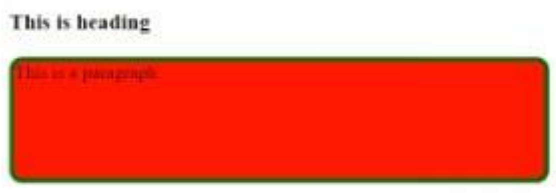
Our basic HTML code goes like this –

```
<body>
    <h3>This is heading</h3>
    <p id="firstPara">This is a paragraph</p>

    <h3>This is second heading</h3>
    <p id="secondPara">This is my second paragraph</p>

    <h3>This is third heading</h3>
    <p id="thirdPara">This is my third paragraph</p>
</body>
```

Let us start by making some changes in the first paragraph. If we write the code as –

```
#firstPara{
    background-color: red;
    height: 100px;
    width:455px;
    border: 4px solid green;
    /* border-width: 4px;
    border-color: green;
    border-style: solid;  */
    border-radius: 11px;
}
```

We will see that the background of the text, will change to red with a height of 100px. Talking about its width, it will also be increased by 455px. Talking about border, we can decide its width, type, color. In the above example we will see a 4px-solid and green color border around the text. Border-radius is used to make the ends of the border curvy. All the changes, you made till now, will look like this –

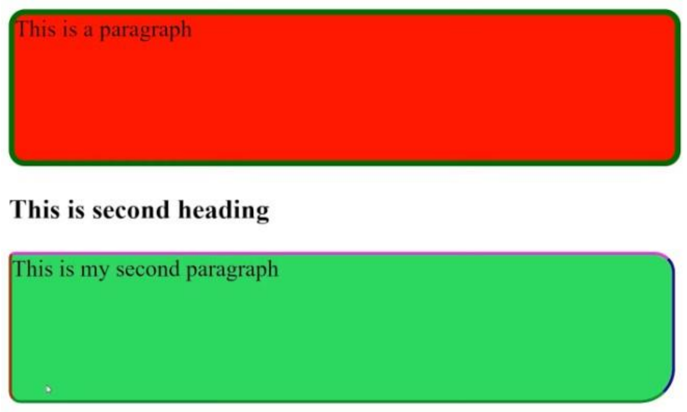

This is heading

This is a paragraph

Now if the condition arises that you want to give a border only at one end, then what will you do. Let us understand with an example -

```
#secondPara{
    background-color: rgb(58, 243, 98);
    height: 100px;
    width:455px;
    border-top: 2px solid rgb(231, 22, 231);
    border-right: 2px solid rgb(18, 10, 133);
    border-bottom: 2px solid rgba(9, 144, 27, 0.774);
    border-left: 2px solid rgb(156, 42, 13);
    border-top-left-radius: 4px;
    border-top-right-radius: 14px;
    border-bottom-left-radius: 8px;
    border-bottom-right-radius: 24px;
}
```

If we want to change the properties of border on the top, it can be done with border-top. Likewise, we can also change the other dimensions with the help of border-right, border- bottom, and border-left as shown in the example. In the same way, we can modify different ends of border with different properties. For example, we can write -border-top-left-radius as 4px, border-top-right-radius as 14px, border-bottom-left-radius as 8px, and border-bottom-right-radius as 24px. All the changes made above, will be shown as-



Now what if, if you want to add a background image behind the text that you have written. Let us understand this with the code –

```
 #thirdPara{
            height: 500px;
            width:455px;
            background-image: url('https://techadroit.com/static/common/img/photo.png');
            border: 2px solid red;
            background-repeat: no-repeat; /* repeat-x and repeat-y will make it repeat on
x and y axis */
            /* background-position: 192px 34px; */
            background-position: center center;
            /* background-position: bottom right; */
            /* background-position: top center; */
          }
```

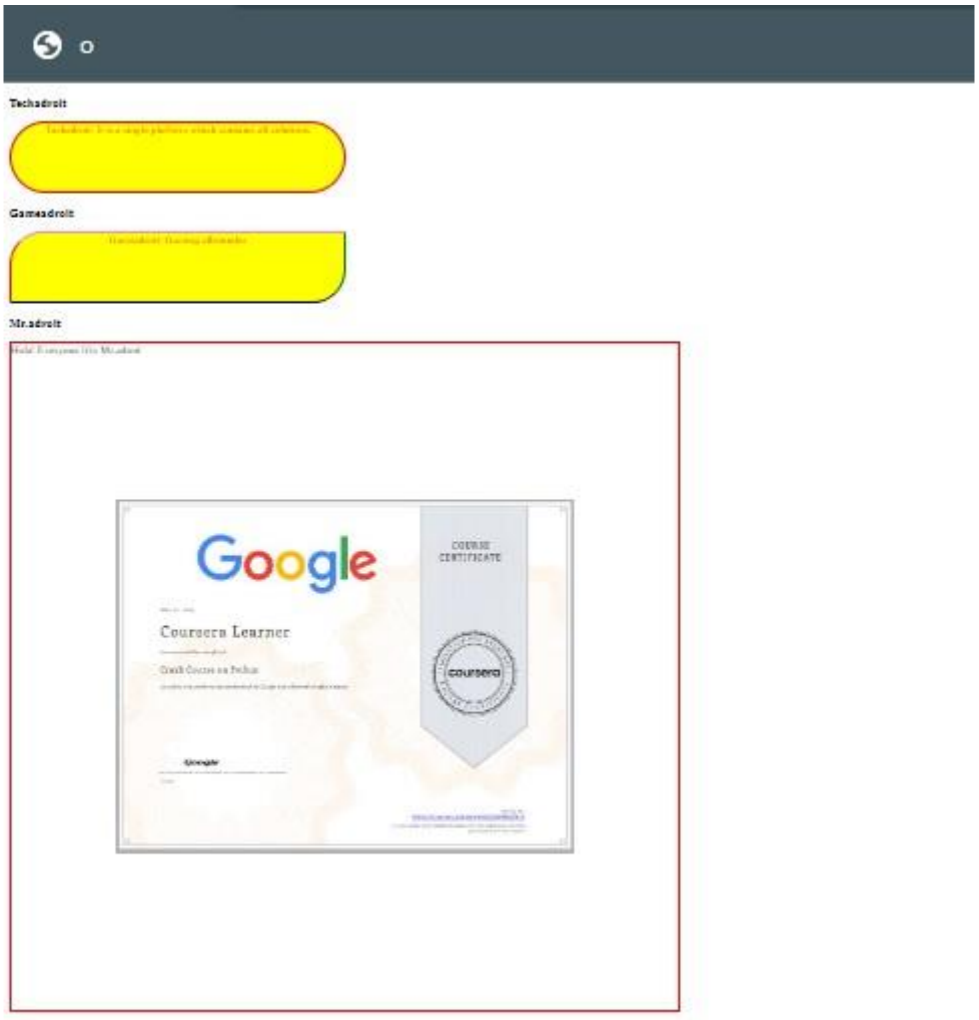
There are two methods for adding a background image. Firstly, you can add directly by adding an URL of the image from website. Secondly, if you are having the files on your local computer, you can directly copy the path of the image. Background position is used to align the image at different positions as per the instructions given.

CODE:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Borders and Backgrounds</title>
    <style>
        #adroit { text-align: center;
            color: red;
            background-color: yellow;
            height: 100px;
            width: 500px;
            border-width: 4px;
            border-color: red;
            border-style: solid;
            border-radius: 130px;
            /* to write border width, color and style in one line */
            /* border: 4px solid red; */
        }
        #adroit2 {
            text-align: center;
            color: red;
            background-color: yellow;
            height: 100px;
            width: 500px;
            border-top: 2px solid violet;
            border-right: 3px solid green;
            border-left: 4px solid red;
            border-bottom: 5px solid black;
            border-top-left-radius: 50px;
            border-bottom-right-radius: 50px;
        }
        #adroit3 {
            height: 1000px;
            width: 1000px;
            background-image: url('certificate.jpg');
            border: 2px solid red;
            /* background-repeat: repeat-x;
            background-repeat: repeat-y; */
            background-repeat: no-repeat;
            /* background-position: center right;
            background-position: top top; */
            background-position: center center;
        }
    </style>
</head>
<body>
    <h3>Techadroit</h3>
    <p id="adroit">Techadroit: It is a single platform which contains all solutions</p>
    <h3>Gameadroit</h3>
    <p id="adroit2">Gameadroit: Gaming allrounder</p>
    <h3>Mr.adroit</h3>
    <p id="adroit3">Hola! Everyone Ii'm Mr.adroit</p>
```

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

Output:



Techadroit

Techadroit: It is a single platform which contains all solutions.

Gameadroit

Gameadroit: Gaming alternative

Mr.adroit

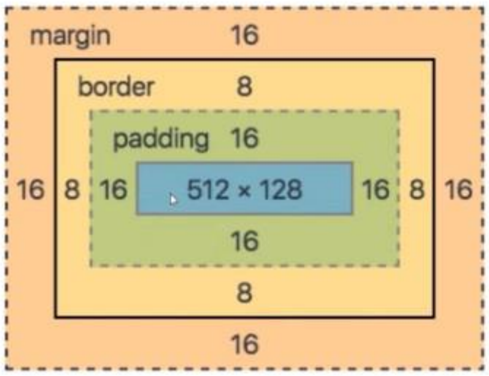
Hello! Everyone It's Mr.adroit

CSS Box Model, Margin and Padding

The box model helps us to define the padding, border, and margin around an element. So from the above diagram we can see where all these things lie around the element. The element is in the center surrounded by **padding, border and margin**.

These parts can be explained as-

- **Content-** The content of the box, where text and images appear.
- **Padding-** It clears an area around the content. The padding is transparent.
- **Border-** A border is one that covers the padding and content.
- **Margin-** It clears an area outside the border. The margin is also transparent.



Let us understand more by writing CSS-

```
.container{
    background-color: rgb(231, 230, 241);
    border: 3px solid rgb(64, 6, 119);

    /* We can set margin/padding for  top, bottom, left and right like this */
    /* padding-top: 79px;
    padding-bottom: 79px;
    padding-left: 34px;
    padding-right: 79px;*/

    /* margin-top: 3px;
    margin-bottom: 5px;
    margin-left: 34px;
    margin-right:5px ; */

    /* margin = top right bottom left;   */
    /* padding = top right bottom left;   */

    /* padding: 23px 56px 6px 78px;    */
    /* margin: 23px 56px 6px 78px;    */

    /* padding: y(top/bottom) x(left/right); */
    /* margin: y(top/bottom) x(left/right); */
    padding: 34px 19px;
    margin: 14px 19px;
    border-radius: 23px;
    width: 533px
}
```

There is padding or margin shorthand for all directions. The first value is for top, 2nd value is for the bottom, 3rd value is for left and 4th value is for right.

```
padding: 23px 56px 6px 78px;
margin: 23px 56px 6px 78px;
```

There is another technique for using the shorthand technique if you want to give the same values for left/right and top/bottom. The first value is the same for both the top and bottom and the second value is the same for both left and right. The two values can be represented as x and y values.

```
padding: 23px 56px;
```

Border radius is used to apply an arc type shape in each corner of the border and its code is written as below:

```
padding: 23px 56px;
```

Let us now understand a property called '**Box sizing'.** On giving width to the element and after that applying padding in the container, the width also changes. It is because in the actual width of an element, margin is already been added into it. If you want this not to happen then you can use the property of 'box-sizing'.

```
box-sizing: border-box;
```

Now if you change the padding then it will adjust the width according to the padding.

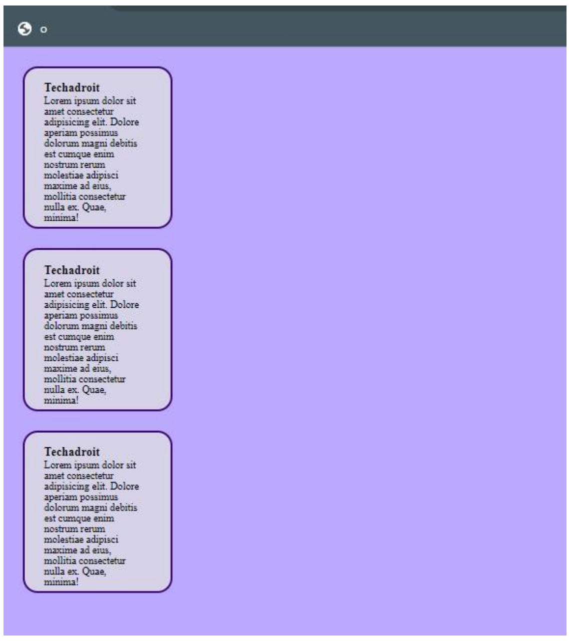
We can take the help of **universal selector** in the CSS to apply the property of box-sizing in all the elements available. It is denoted with a **'*'**.

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}
```

CODE:

```
<!DOCTYPE html>
<html>
<head>
    <title>Box Model</title>
    <style>
        /* universal selector it will work when there is nothing present*/
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
        }
        body {
            background-color: rgb(186, 169, 255);
        }
        .container {
            background-color: rgb(214 211 233);
            border: 3px solid rgb(64, 6, 119);
            padding: 20px 50px 6px 30px; /* padding: top right bottom left*/
            margin: 30px;
            /* padding-top: 30px;
            padding-bottom: 30px;
            margin-top: 20px;
            margin-bottom: 20px; */
            border-radius: 23px;
            width: 233px;
            /* box-sizing: border-box     it is used to adjust width according padding*/
        }
    </style>
</head>
<body>
    <div class="container">
        <h3>Techadroit</h3>
        <p id="adroit">Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolore
aperiam possimus dolorum magni debitis est cumque enim nostrum rerum molestiae adipisci
maxime ad eius, mollitia consectetur nulla ex. Quae, minima!</p>
    </div>

    <div class="container">
        <h3>Techadroit</h3>
        <p id="adroit">Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolore
aperiam possimus dolorum magni debitis est cumque enim nostrum rerum molestiae adipisci
maxime ad eius, mollitia consectetur nulla ex. Quae, minima!</p>
    </div>

    <div class="container">
        <h3>Techadroit</h3>
        <p id="adroit">Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolore
aperiam possimus dolorum magni debitis est cumque enim nostrum rerum molestiae adipisci
maxime ad eius, mollitia consectetur nulla ex. Quae, minima!</p>
    </div>
</body>
</html>
```

Output:



```
Techadroit
Lorem ipsum dolor sit
amet consectetur
adipisicing elit. Dolore
aperiam possimus
dolorum magni debitis
est cumque enim
nostrum rerum
molestiae adipisci
maxime ad eius,
mollitia consectetur
nulla ex. Quae,
minima!

Techadroit
Lorem ipsum dolor sit
amet consectetur
adipisicing elit. Dolore
aperiam possimus
dolorum magni debitis
est cumque enim
nostrum rerum
molestiae adipisci
maxime ad eius,
mollitia consectetur
nulla ex. Quae,
minima!

Techadroit
Lorem ipsum dolor sit
amet consectetur
adipisicing elit. Dolore
aperiam possimus
dolorum magni debitis
est cumque enim
nostrum rerum
molestiae adipisci
maxime ad eius,
mollitia consectetur
nulla ex. Quae,
minima!
```

FLOAT AND CLEAR

The **CSS float** property specifies how an element should float. The **CSS clear** property specifies what elements can float beside the cleared element and on which side. The float property is used for positioning and formatting content, for example, let an image float left to the text in a container. The float property can have one of one of the following values-

- **Left-** The elements floats to the left of its container.
- **Right-** The elements floats to the right of its container.
- **None-** The element does not float (it will be displayed just where it occurs in the text). This is default.
- **Inherit-** The element inherits the float value of its parent.

Let us imagine that we are making a grocery store website and accordingly sell the things.

For the CSS section, we will make different IDs and classes to specify different properties to each item listed. Let us start by defining the classes

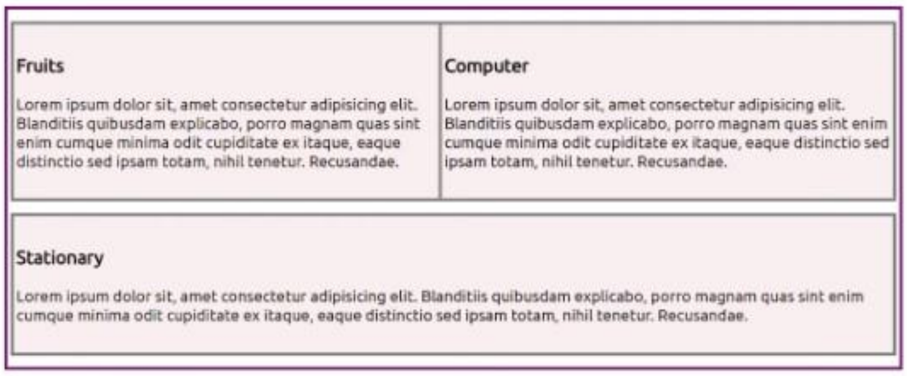
```
.container {
    width: 900px;
    border: 3px solid purple;
    background-color: rgb(250, 226, 205);
    margin: 33px auto;
}

.item {
    border: 3px solid grey;
    margin: 12px 3px;
    padding: 12px 3px;
    background: rgb(248, 238, 238);
}
```

The auto property of margin allows to automatically adjust the margin equally on the both the ends. The result will be as follows as such-

To float the elements, right or left we can target them by their IDs. Let us target all the elements as shown below-

```
#fruit {
            float: right;
            width: 48%;
        }

        #computer {
            float: left;
            width: 48%;
        }

        #stationary {
            /* float: left; */
            clear: both;
            clear: left;
            width: 100%;
        }
```

Initially, if you set the width as 50% for all three, then the result would be as follows-


Fruits

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Blanditiis quibusdam explicabo, porro magnam quas sint enim cumque minima odit cupiditate ex itaque, eaque distinctio sed ipsam totam, nihil tenetur. Recusandae.

Computer

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Blanditiis quibusdam explicabo, porro magnam quas sint enim cumque minima odit cupiditate ex itaque, eaque distinctio sed ipsam totam, nihil tenetur. Recusandae.

Stationary

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Blanditiis quibusdam explicabo, porro magnam quas sint enim cumque minima odit cupiditate ex itaque, eaque distinctio sed ipsam totam, nihil tenetur. Recusandae.

If we add some more texts to fruit and computer and remove the float: left option from stationary then we find that fruit and computer will float on the right side of the container and overlaps the stationary section as follows-

To avoid this, we use the property known as clear. If we write clear: both, then both the other elements will not overlap the stationary section.

For paragraphs, we have different alignments options like right, left, center, and justify. The right alignment will move the texts to the right, left alignment to the left side and so on.

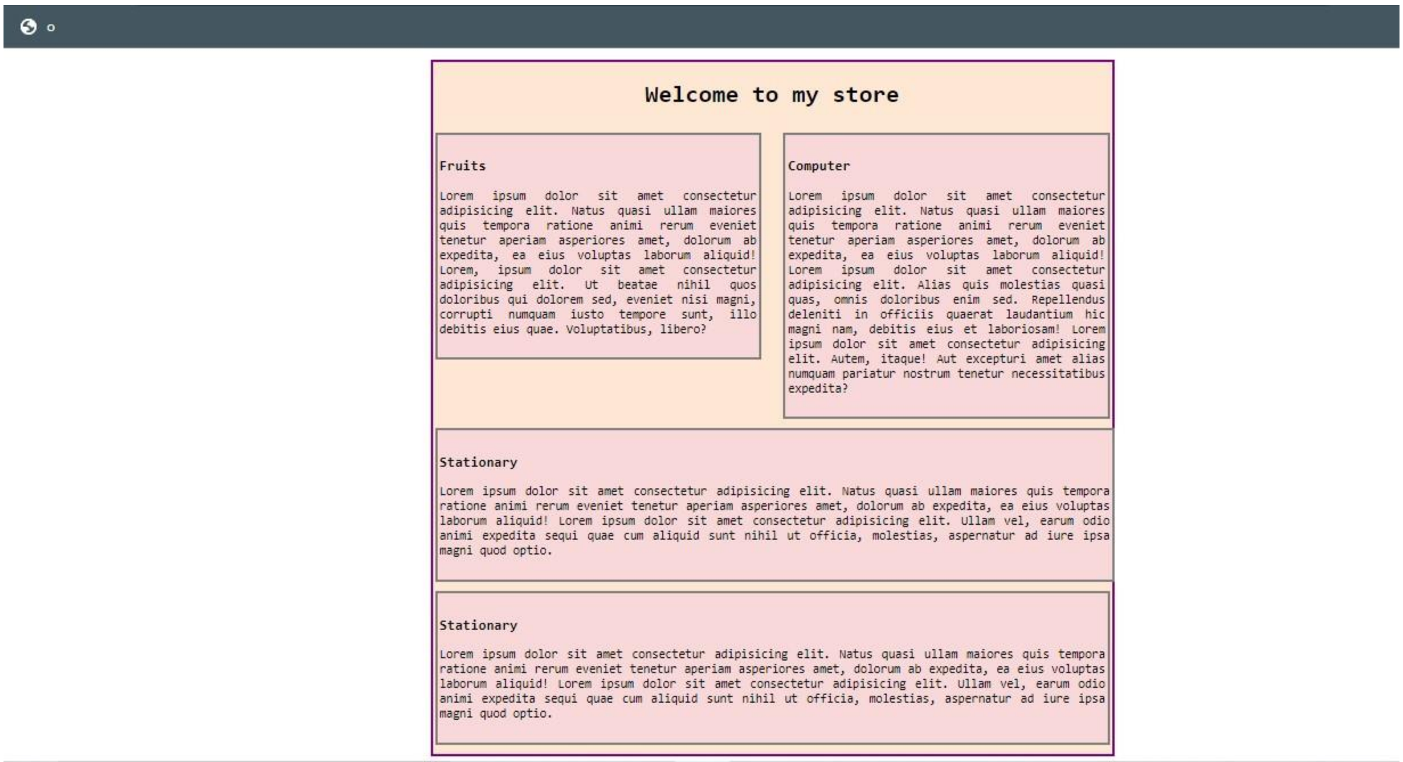
```
 p, h3 {
           /* text-align: right;
           text-align: left;
           text-align: center; */
           text-align: justify;
       }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Alignment</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Inconsolata:wght@300&display=swap');
    </style>
    <style>
        * {
            box-sizing: border-box;
        }

        body {
            font-family: 'Inconsolata', monospace;
        }

        .container {
            width: 900px;
            border: 3px solid purple;
            background-color: rgb(254, 231, 211);
            margin: 50px auto;
        }

        .item {
            border: 3px solid grey;
            margin: 12px 3px;
            padding: 12px 3px;
            background-color: rgb(248, 217, 217);
        }

        #fruits {
            float: left;
            width: 48%;
        }

        #Computer {
            float: right;
            width: 48%;
        }

        #Stationary {
            /* float: left; */
            /* clear: left; */
            /* clear is used to remove overlapping there both is used for left and right
over lapping if the element is left side os we write left on the place of both */
            clear: both;
            width: 100%;
        }

        h3,
        p {
            /* last one command executed like there justy will be wxecute */
            /* text-align: left;
            text-align: right;
            text-align: center; */
            text-align: justify;
        }

        h1 {
            text-align: center;
        }
    </style>
</head>

<body>
    <div class="container">
        <h1>Welcome to my store</h1>
        <div id="fruits" class="item">
            <h3>Fruits</h3>
            <p id="Fruitspara" class="para">Lorem ipsum dolor sit amet consectetur
adipisicing elit. Natus quasi ullam
                maiores quis tempora ratione animi rerum eveniet tenetur aperiam
asperiores amet, dolorum ab expedita,
```

```html
                ea eius voluptas laborum aliquid! Lorem, ipsum dolor sit amet consectetur
adipisicing elit. Ut beatae
                nihil quos doloribus qui dolorem sed, eveniet nisi magni, corrupti numquam
iusto tempore sunt, illo
                debitis eius quae. Voluptatibus, libero?</p>
        </div>
        <div id="Computer" class="item">
            <h3>Computer</h3>
            <p id="Computerpara" class="para">Lorem ipsum dolor sit amet consectetur
adipisicing elit. Natus quasi ullam
                maiores quis tempora ratione animi rerum eveniet tenetur aperiam
asperiores amet, dolorum ab expedita,
                ea eius voluptas laborum aliquid! Lorem ipsum dolor sit amet consectetur
adipisicing elit. Alias quis
                molestias quasi quas, omnis doloribus enim sed. Repellendus deleniti in
officiis quaerat laudantium hic
                magni nam, debitis eius et laboriosam! Lorem ipsum dolor sit amet
consectetur adipisicing elit. Autem,
                itaque! Aut excepturi amet alias numquam pariatur nostrum tenetur
necessitatibus expedita?</p>
        </div>
        <div id="Stationary" class="item">
            <h3>Stationary</h3>
            <p id="Stationarypara" class="para">Lorem ipsum dolor sit amet consectetur
adipisicing elit. Natus quasi
                ullam maiores quis tempora ratione animi rerum eveniet tenetur aperiam
asperiores amet, dolorum ab
                expedita, ea eius voluptas laborum aliquid! Lorem ipsum dolor sit amet
consectetur adipisicing elit.
                Ullam vel, earum odio animi expedita sequi quae cum aliquid sunt nihil ut
officia, molestias, aspernatur
                ad iure ipsa magni quod optio.</p>
        </div>
        <div id="glasses" class="item">
            <h3>Stationary</h3>
            <p id="glassespara" class="para">Lorem ipsum dolor sit amet consectetur
adipisicing elit. Natus quasi ullam
                maiores quis tempora ratione animi rerum eveniet tenetur aperiam
asperiores amet, dolorum ab expedita,
                ea eius voluptas laborum aliquid! Lorem ipsum dolor sit amet consectetur
adipisicing elit. Ullam vel,
                earum odio animi expedita sequi quae cum aliquid sunt nihil ut officia,
molestias, aspernatur ad iure
                ipsa magni quod optio.</p>
        </div>
    </div>
</body>

</html>
```

Output:



Welcome to my store

Fruits

Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus quasi ullam maiores quis tempora ratione animi rerum eveniet tenetur aperiam asperiores amet, dolorum ab expedita, ea eius voluptas laborum aliquid! Lorem, ipsum dolor sit amet consectetur adipisicing elit. Ut beatae nihil quos doloribus qui dolorem sed, eveniet nisi magni, corrupti numquam iusto tempore sunt, illo debitis eius quae. Voluptatibus, libero?

Computer

Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus quasi ullam maiores quis tempora ratione animi rerum eveniet tenetur aperiam asperiores amet, dolorum ab expedita, ea eius voluptas laborum aliquid! Lorem ipsum dolor sit amet consectetur adipisicing elit. Alias quis molestias quasi quas, omnis doloribus enim sed. Repellendus deleniti in officiis quaerat laudantium hic magni nam, debitis eius et laboriosam! Lorem ipsum dolor sit amet consectetur adipisicing elit. Autem, itaque! Aut excepturi amet alias numquam pariatur nostrum tenetur necessitatibus expedita?

Stationary

Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus quasi ullam maiores quis tempora ratione animi rerum eveniet tenetur aperiam asperiores amet, dolorum ab expedita, ea eius voluptas laborum aliquid! Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam vel, earum odio animi expedita sequi quae cum aliquid sunt nihil ut officia, molestias, aspernatur ad iure ipsa magni quod optio.

Stationary

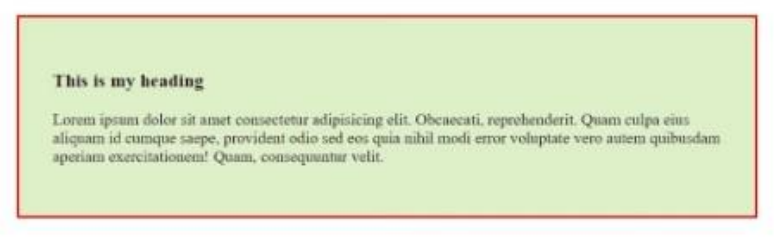
Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus quasi ullam maiores quis tempora ratione animi rerum eveniet tenetur aperiam asperiores amet, dolorum ab expedita, ea eius voluptas laborum aliquid! Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam vel, earum odio animi expedita sequi quae cum aliquid sunt nihil ut officia, molestias, aspernatur ad iure ipsa magni quod optio.

STYLING LINKS & BUTTONS

Let add the basic CSS code to style the HTML part-

```
.container{
            border: 2px solid red;
            background-color: rgb(223, 245, 201);
            padding: 34px;
            margin: 34px auto;
            width: 666px;
        }
```

After writing it, you will observe the changes as follows-



This is my heading

Lorem ipsum dolor sit amet consectetur adipisicing elit. Obcaecati, reprehenderit. Quam culpa eius aliquam id cumque saepe, provident odio sed eos quia nihil modi error voluptate vero autem quibusdam aperiam exercitationem! Quam, consequuntur velit.

We will now design two types of buttons. One will be a normal button and another will be linking to some website. The codes of both are as below-

```
 <a href="https://yahoo.com" class="btn">Read more</a>
<button class="btn">Contact us</button>
```

You will observe that both the buttons will look different. Therefore, to make it look little attractive, we will do some styling in it with CSS.

```css
.btn{
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            font-weight: bold;
            background-color: crimson;
            padding:6px;
            border: none;
            cursor:pointer;
            font-size: 13px;
            border-radius: 4px;
        }
```

To remove the underline in the link part we have to style the anchor tag as-

```css
  a{
            text-decoration: none;
            color: black;
        }
```

Let us now see what Pseudo Selectors are. A pseudo class is used to define a special state of an element.

- **Hover** is used to change the color of text or background of a button as soon as you hover that part. The code for this is as below.

```css
  a:hover{
            color: rgb(5, 0, 0);
            background-color: rgb(221, 166, 38);
        }
```

- The next Pseudo selector is **Visited.** As soon as you visit the anchor tag button and click the link mentioned, it changes its color. To apply this property, write the code as follows-

```css
a:visited{
            background-color: yellow;
        }
```

- The next selector is **Active**. If you visit any button, and click it, it becomes active and showcases with different properties. The code for this is-

```css
  a:active{
            background-color:darkblue;
        }
```

Similarly we can put pseudo selector in the 'btn' class as well. To apply it write the code as follows-

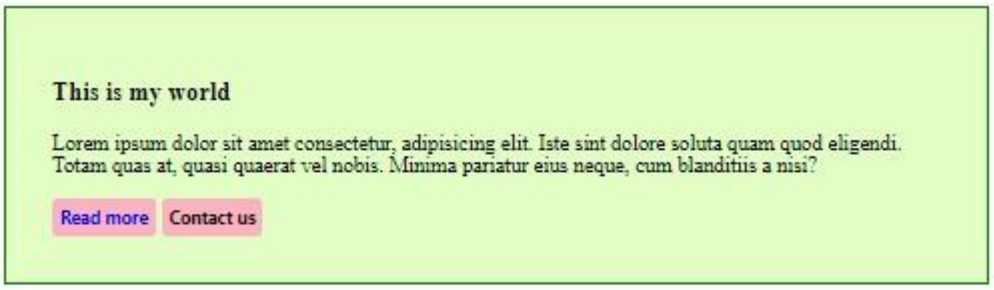
```css
.btn:hover{
            color:darkgoldenrod;
            background-color:rgb(223, 245, 201);
            border: 2px solid black;
        }
```

To learn more about different buttons and pseudo selectors you can visit the website called Bootstrap. There you will find more buttons and properties related to them.

CODE:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Links and Buttons</title>
    <style>
        .container {
            border: 2px solid green;
            background-color: rgb(226, 254, 197);
            padding: 34px;
            margin: 34px auto;
            width: 666px;
        }
        a{
            text-decoration: none; /* it is used to remove the underline from the read 
more */
        }
        /* hover is used to make changes in hyper link */
        a:hover {
            color: black; /* it is used to give the color to the hyper link */
            background-color: green; /* it is used to give the background color to the 
hyper link */
        }

        a:visited {
            background-color: yellow;
        }
        a:active {
            background-color:rgb(251, 179, 194) ;
        }
        .btn {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            font-weight: bold;
            background-color: rgb(251, 179, 194);
            padding: 6px;
            border: none;
            cursor: pointer; /* it is used too change the cursor */
            font-size: 13px;
            border-radius: 4px;
        }
        .btn:hover {
            color: rgb(0, 42, 0);
            background-color: green;
            border: 2px solid black;
        }
    </style>
</head>
<body>
    <div class="container" id="cont1">
        <h3>This is my world</h3>
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Iste sint dolore
soluta quam quod eligendi. Totam quas at, quasi quaerat vel nobis. Minima pariatur eius
neque, cum blanditiis a nisi?</p>
        <a href="https://google.com" class="btn">Read more</a>
        <button class="btn">Contact us</button>
    </div>
</body>
</html>
```

Output:



This is my world

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Iste sint dolore soluta quam quod eligendi. Totam quas at, quasi quaerat vel nobis. Minima pariatur eius neque, cum blanditiis a nisi?

<u>CREATING NAVIGATION MENU</u>

A navigation bar is usually a list of links, so using the <ul> and <li> elements can help in obtaining it. The code for the following will be as follows-

```
 <header>
        <nav class="navbar">
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">About</a></li>
                <li><a href="#">Services</a></li>
                <li><a href="#">Contact us</a></li>
                <div class="search">
                    <input type="text" name="search" id="search" placeholder="Search this
website">
                </div>
            </ul>
        </nav>
    </header>
```

We will now target the navbar class and apply some CSS to make it look more attractive.

First we will change the color of the navigation bar and make its ends circular.

```
.navbar{
            background-color: black;
            border-radius: 30px;
          }
```

```
.navbar{
            background-color: black;
            border-radius: 30px;
          }
```

In the next step, we will make all the nav elements come in single horizontal line

```
  .navbar li{
            float:left;
            list-style: none;
            margin: 13px 20px;
        }
```

The list-style property is used to remove all the bulleted points in the navigation items.

After writing the above code, the background gets removed as it has been overflown by the parent element. To avoid this, we have to write-

```
.navbar ul{
            overflow: auto;
        }
```

Now we will add padding to the all the elements present in the navbar-

```
.navbar li a{
            padding: 3px 3px;
            text-decoration: none;
            color: white;
        }
```

We can also add the search bar in the navigation menu. To do this, we have to write-

```
<div class="search">
                    <input type="text" name="search" id="search" placeholder="Search this
website">
                </div>
```

This will create a search bar in the navigation menu. We can style the search tag by-

```
 .search{
            float: right;
            color: white;
            padding: 12px 75px;
        }
```

We can style the menu available in the navigation bar as-

```
  .navbar input{
            border: 2px solid black;
            border-radius: 14px;
            padding: 3px 17px;
            width: 129px;
        }
```

Within the **'navbar'**, for styling the input tag we can include the border, border-radius, padding, and width as shown above. We can also adjust the padding and other properties using the inspect element on the web page as per your requirements.

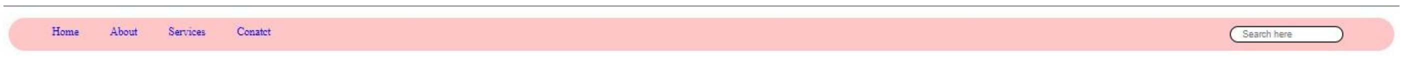
We can also add the hover effect in all the li's. It means whenever we place the pointer on those elements it should change its color.

```
.navbar li a:hover{
            color: red
        }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Navigation</title>
    <style>
        .navbar {
            background-color: rgb(254, 199, 199);
            border-radius: 40px;
        }

        .navbar ul {
            overflow: auto;
        }

        .navbar li {
            float: left;
            list-style: none;
            margin: 13px 20px;

        }

        .navbar li a {
            padding: 3px 3px;
            text-decoration: none;
        }

        .navbar li a:hover {
            color: red;
        }
        .search {
            float: right;
            color: white;
            padding: 12px 75px;
        }

        .navbar input {
            border: 2px solid black;
            border-radius: 14px;
            padding: 3px 17px;
            width: 129px;
        }
    </style>
</head>

<body>
    <header>
        <nav class="navbar">
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">About</a></li>
                <li><a href="#">Services</a></li>
                <li><a href="#">Conatct</a></li>
                <div class="search">
                    <input type="text" name="search" id="search" placeholder="Search
here">
                </div>
            </ul>
        </nav>
    </header>
</body>

</html>
```

Output:



CSS DISPLAY PROPERTY

Let us style the image and heading with some CSS-

```
  img {
            margin: auto;
            display: block;
            width: 34px;
        }

        h3 {
            text-align: center;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0px;
        }
```

By inspecting both the elements in the Chrome browser, we see that the image is an inline element and the h3 heading is the block element. Our objective is to bring all the elements to the center of the webpage. We can achieve it by adjusting the width of the block element i.e. the heading. The respective code of the following is-

```
 header {
            border: 2px solid red;
            margin: auto;
    width: 1200px;
        }
```

The display of "img" is inline and therefore, to make it come to center, we have to set the property display as block as follows-

```
img {
            margin: auto;
            display: block;
            width: 34px;
        }
```

The next problem which arises is that when we stretch the full width of the page, the text in the heading moves towards left. So to move it towards the center, we can set the property of **text-alignment as center.**

Display inline means it will take the space according to the size of the element. **Display block** means we can set its width and by margin manually.

Now suppose we want to make an element inline as well as customize its width too, then in that case we can **use inline-block**. To understand it, first we will add three divs with some texts in it and then style it. To appear those as a box, we can take the help of container and box class. We can style the box element as

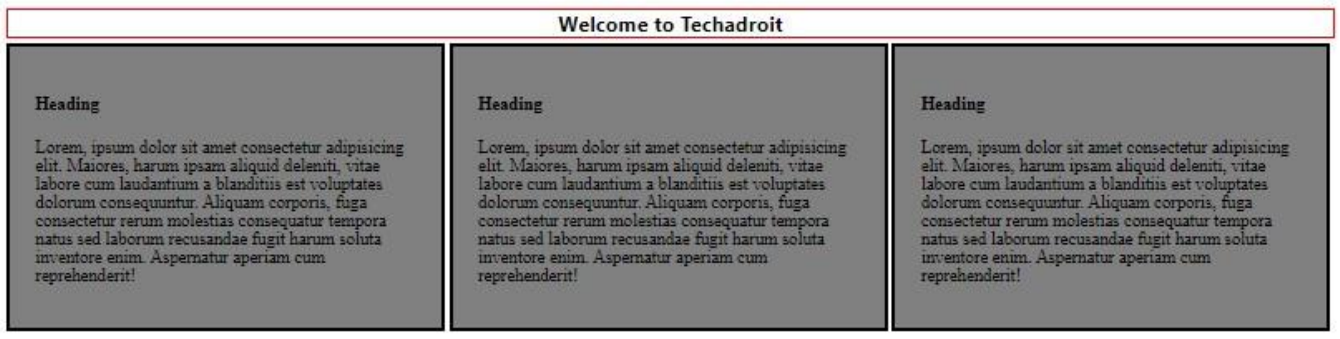
```
.box {
            border: 4px solid black;
            background-color: grey;
            margin: 4px 0px;
            padding: 23px;
            width: 33%;
            box-sizing: border-box;
            display: inline-block;
        }
```

The inline-block property here allows us to change the width of inline elements also. To ensure that all the three blocks come in a single line, we can use the property box-sizing. It ensure that the width we provide is not changed including padding and margin.

CODE:

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>CSS Display property</title>
    <style>
        * {
            box-sizing: border-box;
        }

        header {
            border: 2px solid red;
            margin: auto;
            width: 1200px;
        }

        img {
            margin: auto;
            display: block;
            width: 34px;
        }

        h3 {
            text-align: center;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0px;
        }

        .box {
            border: 4px solid black;
            background-color: grey;
            margin: 4px 0px;
            padding: 23px;
            width: 33%;
            box-sizing: border-box;
            display: inline-block;
        }

        .container {
            margin: auto;
            width: 1200px;
        }
    </style>
</head>

<body>
    <header class="top">
        <img src="gojo.png" alt="">
        <h3>Welcome to Techadroit</h3>
    </header>
    <div class="container">
        <div class="box">
            <h4 class="heading">Heading</h4>
            <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Maiores, harum
ipsam aliquid deleniti, vitae
                labore cum laudantium a blanditiis est voluptates dolorum consequuntur.
Aliquam corporis, fuga
                consectetur rerum molestias consequatur tempora natus sed laborum
recusandae fugit harum soluta
                inventore enim. Aspernatur aperiam cum reprehenderit!</p>
        </div>
        <div class="box">
            <h4 class="heading">Heading</h4>
            <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Maiores, harum
ipsam aliquid deleniti, vitae
                labore cum laudantium a blanditiis est voluptates dolorum consequuntur.
Aliquam corporis, fuga
                consectetur rerum molestias consequatur tempora natus sed laborum
recusandae fugit harum soluta
                inventore enim. Aspernatur aperiam cum reprehenderit!</p>
        </div>
        <div class="box">
            <h4 class="heading">Heading</h4>
            <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Maiores, harum
ipsam aliquid deleniti, vitae
                labore cum laudantium a blanditiis est voluptates dolorum consequuntur.
Aliquam corporis, fuga
                consectetur rerum molestias consequatur tempora natus sed laborum
recusandae fugit harum soluta
                inventore enim. Aspernatur aperiam cum reprehenderit!</p>
        </div>
    </div>
</body>

</html>
```

Output:



Welcome to Techadroit

Heading

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Maiores, harum ipsam aliquid deleniti, vitae labore cum laudantium a blanditiis est voluptates dolorum consequuntur. Aliquam corporis, fuga consectetur rerum molestias consequatur tempora natus sed laborum recusandae fugit harum soluta inventore enim. Aspernatur aperiam cum reprehenderit!

Heading

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Maiores, harum ipsam aliquid deleniti, vitae labore cum laudantium a blanditiis est voluptates dolorum consequuntur. Aliquam corporis, fuga consectetur rerum molestias consequatur tempora natus sed laborum recusandae fugit harum soluta inventore enim. Aspernatur aperiam cum reprehenderit!

Heading

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Maiores, harum ipsam aliquid deleniti, vitae labore cum laudantium a blanditiis est voluptates dolorum consequuntur. Aliquam corporis, fuga consectetur rerum molestias consequatur tempora natus sed laborum recusandae fugit harum soluta inventore enim. Aspernatur aperiam cum reprehenderit!

Position absolute, relative, fixed and sticky

Types Of Position Property :

There are five types of position property :

- static
- relative
- absolute
- fixed
- sticky

position: static;

- It is the default position of HTML elements.

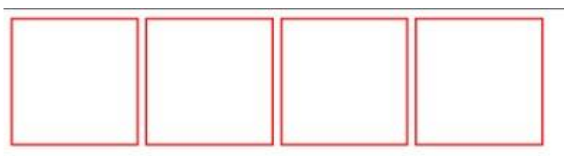
position: relative;

- It is used when we need to position an HTML element relative to its normal position.
- We can set the top, right, bottom, and left properties that will cause the element to adjust away from the normal position.

Example: We have used the below CSS to design four boxes as shown in the given image :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Position Tutorial</title>
    <style>
        .box{
            border: 2px solid red;
            display: inline-block;
            width: 150px;
            height: 150px;
            margin: 2px;
        }
    </style>
</head>
<body>
    <div class="box" id="box1"></div>
    <div class="box" id="box2"></div>
    <div class="box" id="box3"></div>
    <div class="box" id="box4"></div>
</body>

</html>
```

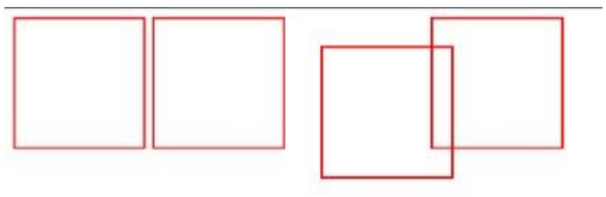
Output:



The default position of all the boxes in the above image is static. Now, we will change the position from static to relative of box 3. Here is the CSS used :

```
<style>
       .box {
           border: 2px solid red;
           display: inline-block;
           width: 150px;
           height: 150px;
           margin: 2px;
       }
       #box3 {
           position: relative;
           top: 34px;
           left: 34px;
 }
</style>
```

You can see in the image given below that box3 has shifted 34px away from the top and left side relative to its normal position.



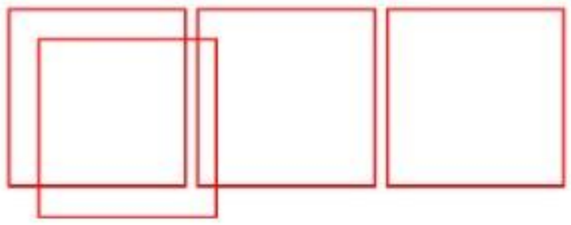
position: absolute;

- An element with the absolute position will move according to the position of its parent element.
- In the absence of any parent element, the HTML element will be placed relative to the page.

Now, we have changed the position of box3 from relative to absolute. Here is the CSS used :

```
<style>
       .box {
           border: 2px solid red;
           display: inline-block;
           width: 150px;
           height: 150px;
           margin: 2px;
       }
       #box3 {
           position: absolute;
           top: 34px;
           left: 34px;
           }
       .container{
           border: 2px solid black;
           background-color: khaki;
           height: 3444px;
       }
    </style>
```

You can see in the image given below that the box3 has moved to the left side of the page.



position: fixed;

- An element with position:fixed; will remain stuck to a specific position even after the page is scrolled.
- This position property is used when we want to keep an HTML element at a fixed spot no matter where on the page the user is.

Notice the box fixed at the top right corner of the page in the image given below. Here is the CSS used :

```
 <!DOCTYPE html>

<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fixed Position In CSS</title>
    <style>
        .box{
            border: 2px solid red;
            display: inline-block;
            width: 150px;
            height: 150px;
            margin: 2px;
        }
        #box3{
            position: fixed;
            right: 4px;
        }
    </style>
</head>
<body>
    <div class="box" id="box1"></div>
    <div class="box" id="box2"></div>
    <div class="box" id="box3"></div>
    <div class="box" id="box4"></div>
</body>
</html>
```

position: sticky;

- It is a hybrid of relative and fixed position.
- An HTML element with position:sticky; will just sit there until a given position offset is met.

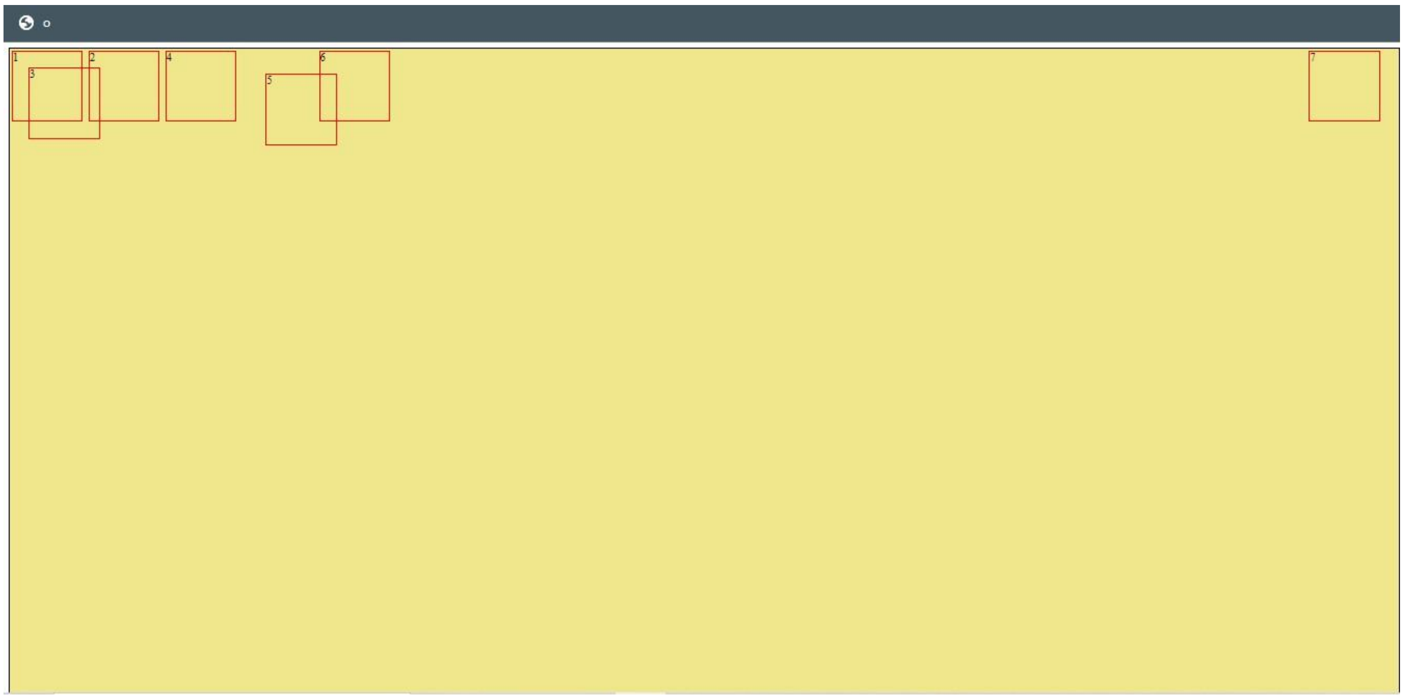
Use the CSS given below to get a better understanding of the sticky element.

```
 #box3 {
        position: sticky;
        top: 3px;
            }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Positions</title>
    <style>
        .container{
            border: 2px solid black;
            background-color: khaki;
            /* margin: 50px 0px 50px; */
            height: 1000px;

        }

        .box{
            display: inline-block;
            border: 2px solid red;
            width: 100px;
            height: 100px;
            margin: 2px;
            /* float: left; */
        }
        #box3{
            /* absolute position : relative to its parent class (there the parent class is
container for the box1,2,3,4,5, and 6) */
            position: absolute;
            top: 34px;
            left: 34px;
        }
        #box7{
            /* fixed position : It will fixed the position ( means the position will not
be changed even if we are scrolling) */
            position: fixed;
            right: 34px;
        }
        #box5{
            /* relative position : relative to its original position and will leave a gap
at its original position */
            position: relative;
            top: 34px;
            left: 34px;
        }
        #box4{
            /* sticky position : It used to stick the element which will not be moved even
we are scrolling  */
            position: sticky;
            top: 3px;
        }

    </style>
</head>
<body>
    <!-- <h3>Relative position</h3> -->
    <div class="container">
        <div class="box" id="box1">1</div>
        <div class="box" id="box2">2</div>
        <div class="box" id="box3">3</div>
        <div class="box" id="box4">4</div>
        <div class="box" id="box5">5</div>
        <div class="box" id="box6">6</div>
        <div class="box" id="box7">7</div>
    </div>

</body>
</html>
```

Output:



PROJECT : 1

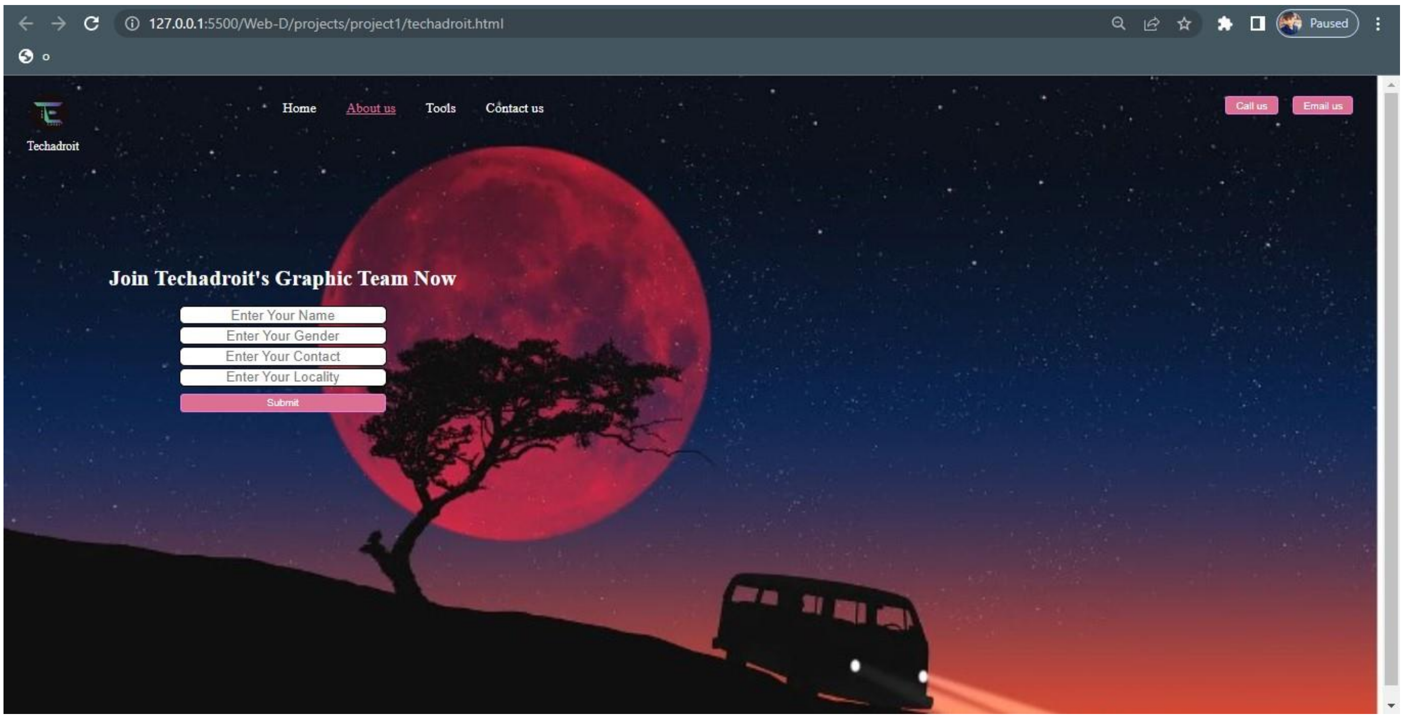
CREATING A WEBSITE USING HTML5 & CSS3

CODE:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Techadroit's Graphic</title>
    <link rel="stylesheet" href="style.css">
    <style>
        /* CSS Reset  */
        body {
            /* font-family: 'Baloo Bhai', cursive; */
            margin: 0px;
            padding: 0px;
            height: 100vh;
            width: 200px;
            background-size: cover;
            background-repeat: no-repeat;
            background-image: url(img/ed.jpg);
            color: white;
        }
        .left{
            display: inline-block;
            /* border: 2px solid red; */
            position: absolute;
            left: 34px;
            top: 20px;

        }
        .left img{
            width: 70px;
            /* changing the color of logo from black to white by using filter function  */
            /* filter: invert(10%) */


        }
        .left div{
            line-height: 19px;
            text-align: center;
            margin: auto;
            font-size: 18px;
        }
        .mid{
            display: block;
            width: 400%;
            margin: 20px 350px;
            /* border: 2px solid green; */
```

```css
        }
        .right{
            position: absolute;
            right: 34px;
            top: 30px;
            display: inline-block;
            /* border: 2px solid yellow; */
        }
        .navbar{
            display: inline-block;
        }
        .navbar li{
            display: inline-block;
            list-style: none;
            font-size: 20px;
        }
        .navbar li a{
            color : white;
            text-decoration: none;
            padding: 30px 20px;
        }
        .navbar li a:hover, .navbar li a:active{
            color : palevioletred;
            text-decoration: underline;

        }
        .btn{
            margin: 0px 9px;
            background-color: palevioletred;
            color: white;
            padding: 4px 14px;
            border: 2px solid violet;
            border-radius: 5px;
            font-size: 15px;
            cursor: pointer;
        }
        .btn:hover{
            background-color: rgb(236, 153, 201);
        }
        .container{
            /* border: 2px solid red; */
            margin: 106px 80px;
            padding: 75px ;
            width: 255%;
            border-radius: 28px;

        }
        .form-group input{
            text-align: center;
            display: block;
            width: 300px;
            padding: 1px;
            border: 2px solid black;
            margin: 3px auto;
            font-size: 20px;
            border-radius: 8px;
        }
        .container button{
            display: block;
            width: 59%;
            margin: 10px auto;
        }
    </style>
</head>
<body>
    <header>
        <!-- left box for logo  -->
        <div class="left">
            <img src="img/logo.png" alt="Remote logo">
            <div>Techadroit</div>
        </div>
        <!-- mid box for navbar  -->
        <div class="mid">
            <ul class="navbar">
                <li><a href="#">Home</a></li>
                <li><a href="#">About us</a></li>
                <li><a href="#">Tools</a></li>
                <li><a href="#">Contact us</a></li>
            </ul>
        </div>
        <!-- right box for buttons  -->
        <div class="right">
            <button class="btn">Call us</button>
            <button class="btn">Email us</button>
        </div>
    </header>
    <div class="container">
        <h1>Join Techadroit's Graphic Team Now</h1>
        <form action="backend.php">
            <div class="form-group">
                <input type="text" name="name" id="" placeholder="Enter Your Name">
            </div>
            <div class="form-group">
                <input type="text" name="name" id="" placeholder="Enter Your Gender">
            </div>
            <div class="form-group">
                <input type="text" name="name" id="" placeholder="Enter Your Contact">
            </div>
            <div class="form-group">
                <input type="text" name="name" id="" placeholder="Enter Your Locality">
            </div>
            <button class="btn">Submit</button>
        </form>
    </div>
</body>
</html>
```

Output:



VISIBILITY & Z-INDEX

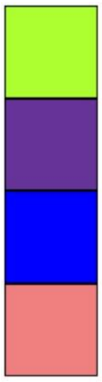
CSS Visibility Property:

First of all, let's create four boxes of different colours. Here is the CSS used :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Visibility and Z-Index Property In CSS</title>
    <style>
        .box {
            width: 170px;
            height: 170px;
            border: 2px solid black;
        }
        #box1 {
            background-color: greenyellow;
        }
        #box2 {
            background-color: rebeccapurple;
        }
        #box3 {
            background-color: blue;
        }
        #box4 {
            background-color: lightcoral;
        }
    </style>
</head>
```

```
<body>
    <div class="box" id="box1"></div>
    <div class="box" id="box2"></div>
    <div class="box" id="box3"></div>
    <div class="box" id="box4"></div>
</body>
</html>
```

Output:



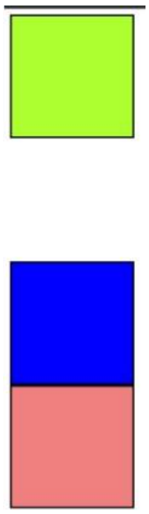
Now, let's start our discussion on visibility property :

- Visibility property is used to hide or show an HTML element without changing the layout of the page.
- The hidden element uses the space on the page because it is still there, but it is not visible to the user.

Now, we will change the visibility from visible to hidden of box2 for a better understanding of visibility property. Here is the CSS used :

```
<style>
        .box {
            width: 170px;
            height: 170px;
            border: 2px solid black;
        }
        #box1 {
            background-color: greenyellow;
        }
        #box2 {
            background-color: rebeccapurple;
            visibility: hidden;
        }
        #box3 {
            background-color: blue;
        }
        #box4 {
            background-color: lightcoral;
        }
```

You can see in the image given below that the box2 is not visible anymore, but it is still occupying the space on the page.



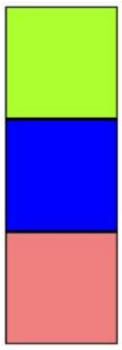
Difference Between display:none; and visibility:hidden;

There is a minor difference between the display: none; and visibility:hidden; property of CSS. Let's understand this difference with the help of the boxes that we created earlier.

Use the following CSS for box2 :

```
#box2 {
            background-color: rebeccapurple;
            display: none;
        }
```

In the above code, we have changed the display value of box2. Here are the results :



In the above image, you can clearly see that box2 is completely removed from the webpage, and there is no empty space left on the page. But, when we used the visibility: hidden property, the box2 element was still occupying the space.

- display:none; - It completely removes an HTML tag from the web page like it was never there.
- visibility:hidden; - It makes the tag invisible but will not remove the element, and it will still occupy the space on the page.

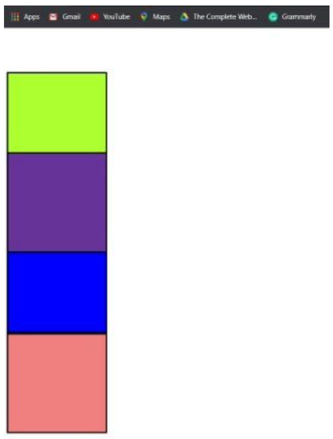
Z-Index Property In CSS :

At the starting of this tutorial, we created four boxes of different colours. Now, try to answer this question: What if one box overlaps the other? Which box will be visible to the user? This is where z-index property comes into the picture. So, whenever HTML elements collapse with each other, then the element with smaller z-index value will be covered by the element with larger z-index value.

Note: Z-index does not work on static position value. It only works on the elements with position: relative, absolute, fixed, or sticky. We are changing the positions of box1 and box2 by applying the CSS given below:

```
#box1 {
          top: 69px;
          position: relative;
          background-color: greenyellow;
      }
      #box2 {
          top: 34px;
          position: relative;
          background-color: rebeccapurple;
      }
```

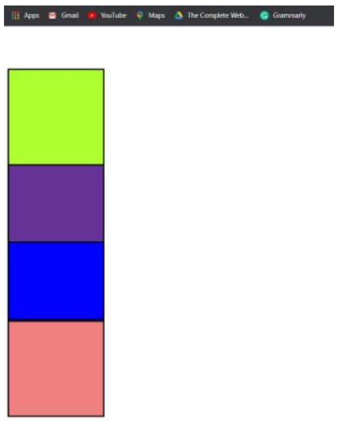
Output:



You can see in the above image that the box2 overlaps the box1. Now, we will give z-index value to box1 and box2. Here is the CSS used :

```
     #box1 {
          top: 69px;
          position: relative;
          background-color: greenyellow;
          z-index: 1;
      }
      #box2 {
          top: 34px;
          position: relative;
          background-color: rebeccapurple;
          z-index: 0;
      }
```

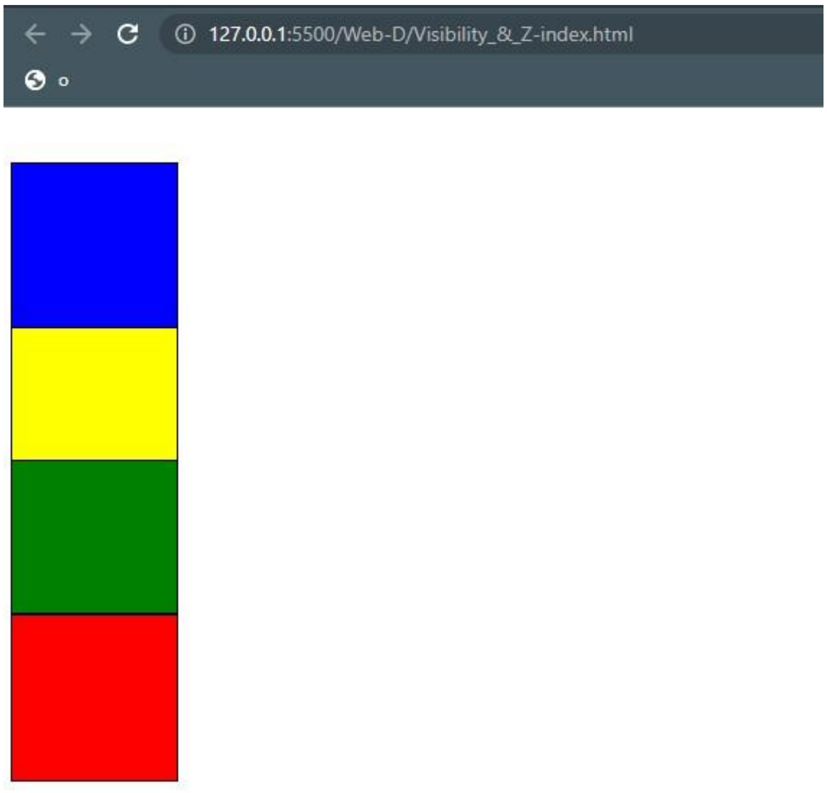
In the above code, we have set the z-index value of box1 and box 2, respectively. Here are the results :



CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Visibility and Z-index</title>
    <style>
        .box{
            width: 170px;
            height: 170px;
            border: 2px solid black;
        }
        #box1{
            position: relative;
            z-index: 34;
            background-color: blue;
            top: 49px;
        }
        #box2{
            background-color: yellow;
            /* display: none;   -  will hide the element and the space  */
            /* visibility: hidden;   - will hide the element but will show its empty
space  */
            top: 14px;
            position: relative;
            z-index: 32;
        }
        #box3{
            background-color: green;
        }
        #box4{
             background-color: red;
            }
    </style>
</head>
<body>
    <div class="box" id="box1"></div>
    <div class="box" id="box2"></div>
    <div class="box" id="box3"></div>
    <div class="box" id="box4"></div>
</body>
</html>
```

Output:


127.0.0.1:5500/Web-D/Visibility_&_Z-index.html

FLEXBOX

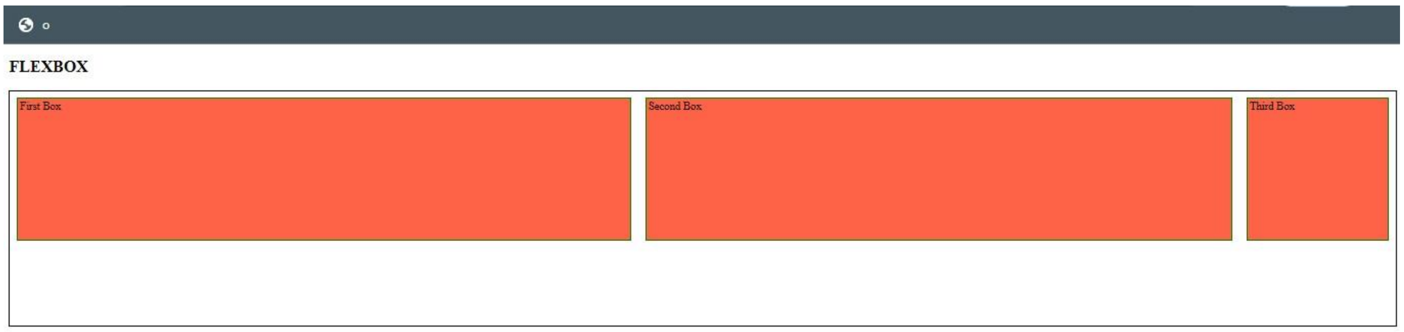
The Flexbox Module was designed as a one-dimensional layout model, and as a method that could offer space distribution between items in an interface with powerful alignment capabilities.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flexbox</title>
    <style>
        .container{
            height: 344px;
            width: 100%;
            border: 2px solid black;
            /* initialize the container as a flex box  */
            display: flex;
            /* flex properties for flex container  */
            /* flex-direction: row;  row is by default */
            flex-direction: row;
            /* flex-direction: column;
            flex-direction: row-reverse;
            flex-direction: column-reverse; */
            /* flex-wrap: wrap;   It is used to control the flex container is single line
or multi lines */
            flex-wrap: wrap;
            /* flex-flow: row-reverse wrap; It is used to give both flex direction and
flex wrap at once  */
            /* justify-content: center;
            justify-content: space-between;
            justify-content: space-evenly; */
            /* align-items: center;
            align-items: flex-end;
            align-items: flex-start;
            align-items: stretch; */
        }
        .item{
            width: 200px;
            height: 200px;
            background-color: tomato;
            border: 2px solid green;
            margin: 10px;
            padding: 3px;
        }
        #item-1{
            /* flex properties for a flex item  */
            /* order is used to give the order to the items of flex the highest order item
will be at the end of the flex  */
            /* order: 2;  */
            /* flex-grow is used to give the space to the item  */
            /* in this flex grow it will take space of 2 items space  */
            flex-grow: 2;
            flex-shrink: 2;


        }
        #item-2{
            flex-grow: 2;
            flex-shrink: 3;
            flex-basis: 160px;
            /* when flex-direction is set to row then flex-basis will control width  */
            /* when flex-direction is set to column then flex-basis will control
height  */

        }
        #item3{
            /* order: 40; */
            /* flex: 2 3 160px;  */
            align-self: flex-start;
            align-self: flex-end;
            align-self: center;

        }
    </style>
```

```
</head>
<body>
    <h2>FLEXBOX</h2>
    <div class="container">
        <div class="item" id="item-1">First Box</div>
        <div class="item" id="item-2">Second Box</div>
        <div class="item" id="item-3">Third Box</div>
        <!-- <div class="item" id="item-4">Fourth Box</div>
        <div class="item" id="item-5">Fifth Box</div>
        <div class="item" id="item-6">Sixth Box</div> -->
    </div>
</body>
</html>
```

Output:



em, rem, vh and vw units + Responsive design

What Is Responsive Design?

Have you ever noticed that websites nowadays adjust themselves according to the resolution of your device(Smartphone, tablet, or desktop computer)? Isn't it amazing that a website is automatically changing its height and width according to the size of the user's screen? This is possible because of the responsive design. Let's dive deep into responsive design.

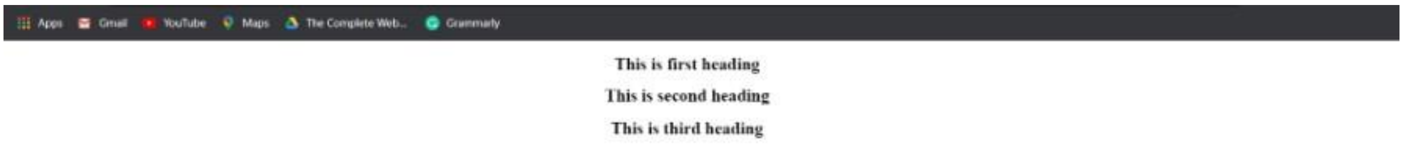
- Responsive design is a way for a web developer to make a website adapt to all the devices and resolutions.
- Endless new resolutions and devices are challenging to support separately for a web developer; therefore, responsive design is the best approach to develop a website that can adjust itself according to the screen size.
- Responsive design is a necessity and not a luxury anymore!

Various Ways To Achieve Responsive Design :

- By using rem/vh/vw units over pixels.
- By using max-width/max-height.
- Using CSS Media Queries.
- Setting up the viewport.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Size Units</title>
    <style>
        h1{
            text-align: center;
            font-size: 10px;
        }
        .container{
            font-size: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
   <h1 id="first">This is first heading</h1>
   <h1 id="second">This is second heading </h1>
   <h1 id="third">This is third heading</h1>
</div>
</body>
</html>
```

Output:



```
This is first heading
This is second heading
This is third heading
```

Here, we have just created three headings and aligned them to the center. Now, we will use these heading to understand the concept of em, rem,vh, and vw.

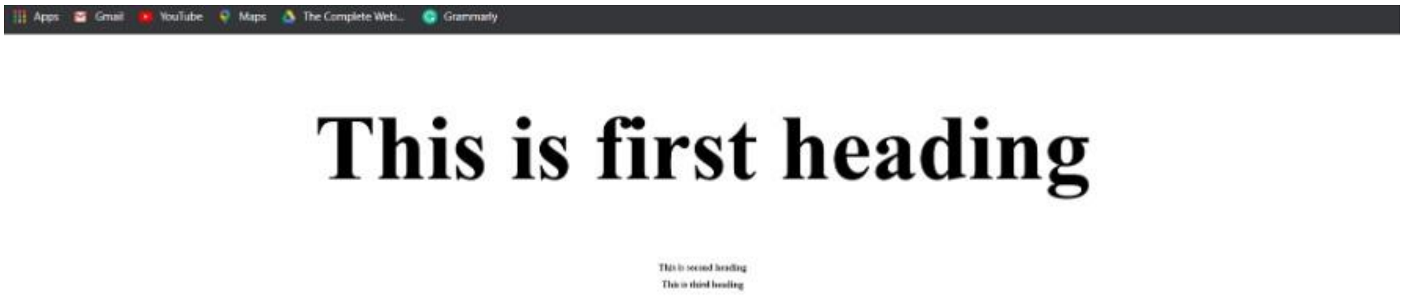
em-

- Font-sizes are inherited relative to the parent element.
- 10em means ten times of the parent element.

Use the CSS given below :

```
 <style>
        h1{
            text-align: center;
            font-size: 10px;
        }
        .container{
            font-size: 10px;
        }
        #first{
            font-size: 10em;
        }
</style>
```

Output:



This is first heading

This is second heading

This is third heading

In the above image, you can see that the font-size of the first heading is changed. Earlier the font-size was 10px because <h1> inherited this size from its parent element, i.e., container. Now, the font-size of <h1> has changed to 100px because 10 em means ten times of the parent element so 10*10=100px.

rem-

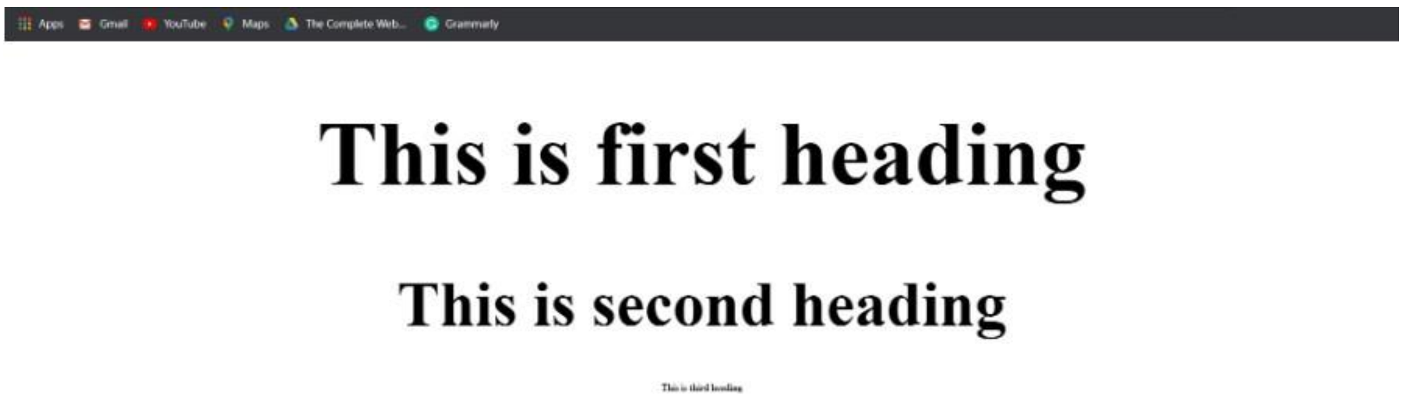
- Font-size gets set according to the font-size of the root element.
- In general, <html> is the root element.
- In rem, "r" stands for "root."

Use the CSS given below:

```
<style>
        html{
            font-size: 7px;
        }
        h1{
            text-align: center;
            font-size: 10px;
        }
        .container{
            font-size: 10px;
        }
        #first{
            font-size: 10em;
        }
        #second{
            font-size: 10rem;
        }
    </style>
```

In the above code, we have given the font-size of 7px to the <html>. Then, we have applied the font-size of 10rem to the second heading. Here is the output :

Output:



This is first heading

This is second heading

This is third heading

In the above image, you can see that the font size of the second heading has changed from 7px to 70px because 10rem is equal to 10 times the root element. You can verify the font-size with the help inspect element functionality of the chrome browser.

vh-

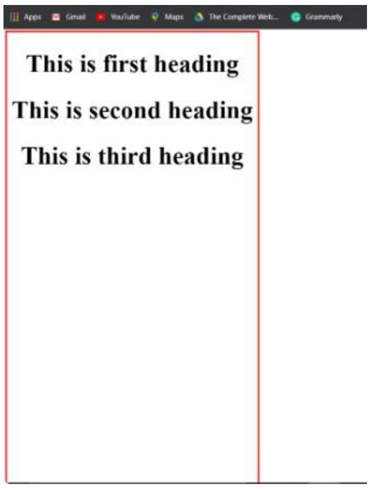
- It stands for viewport height.
- vh is equal to the 1/100 times of the viewport height.
- Example: Suppose height of the browser is 1000px, so 1vh is equaled to (1/100)*1000= 10px.

Use the CSS given below :

```
<style>
        html{
            font-size: 7px;
        }
        h1{
            text-align: center;
            font-size: 40px;
        }
        .container{
          border: 2px solid red;
          height: 100vh;
          width: 400px;
        }
        /* #first{
            font-size: 10em;
        }
        #second{
            font-size: 10rem;
        } */
    </style>
```

In the above code, we have made a border and assigned a height of 100vh to it. Here are the results :

Output:


This is first heading

This is second heading

This is third heading

In the above image, you can see that the border's height is changed to 100% of the viewport.

vw-

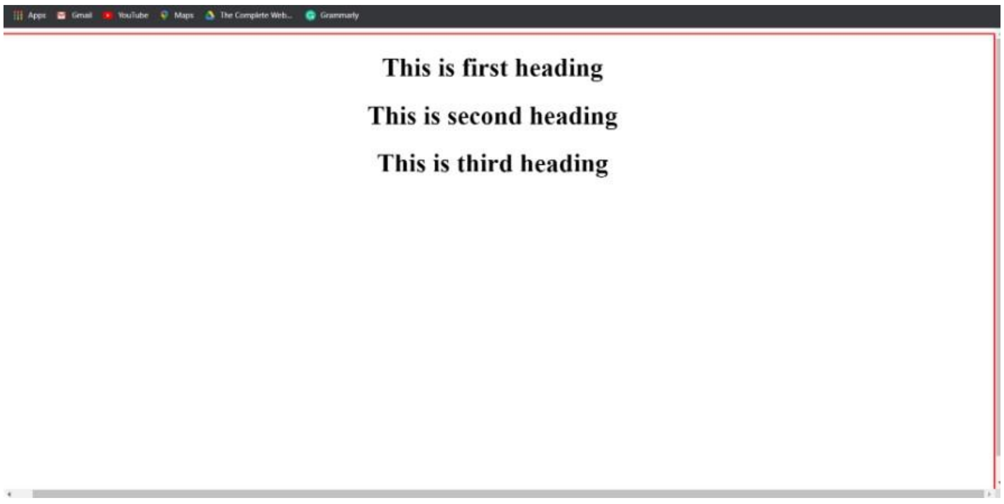
- It stands for viewport width.
- Similar to vh, vw is equal to the 1/100 times of the viewport width.
- Example: Suppose width of the browser is 1000px, so 1vw is equaled to (1/100)*1000= 10px.

Use the CSS given below

```
<style>
        html{
            font-size: 7px;
        }
        h1{
            text-align: center;
            font-size: 40px;
        }
        .container{
          border: 2px solid red;
          height: 100vh;
          width: 100vw;
        }
        /* #first{
            font-size: 10em;
        }
        #second{
            font-size: 10rem;
        } */
    </style>
```

In the above code, we have assigned a width of 100vw to the border. Here are the results:

Output:



This is first heading

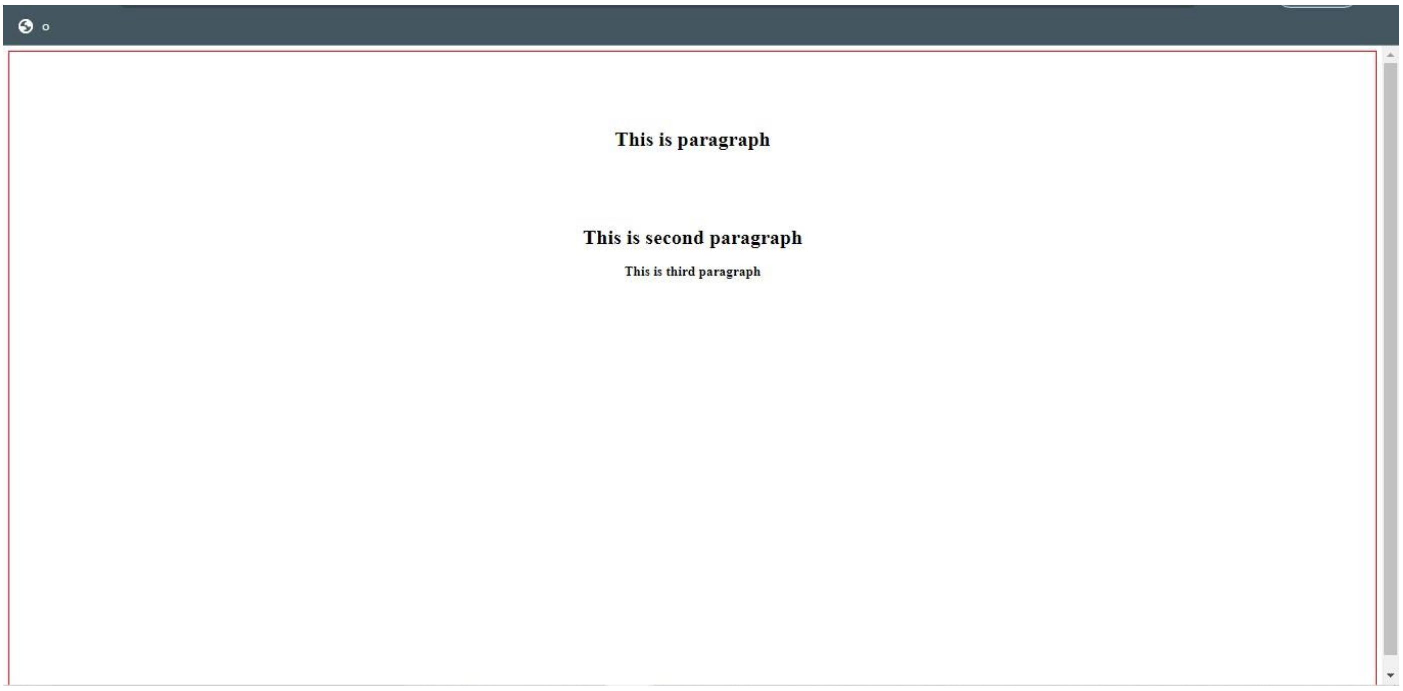
This is second heading

This is third heading

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Desing and Units</title>
    <style>
        .container{
            /* width: 400px; */
            /* wh is used to takke the full width of the window  */
            width: 100wh;
            /* height: 344px; */
            /* vh is used to take the full height of the window  */
            height: 100vh;
            font-size: 10px;
            border: 2px solid red;
        }
        html{
            font-size: 10px;
        }
        h1{
            text-align: center;
        }
        #first{
            /* in font-size em will increase the font size 3 times to its parent, means it
will be 30px */
            font-size: 3em;
            /* in padding it will not follow the parent it will increase 3 times of the
font size, means it will be 90px  */
            padding: 3em;
        }
        #second{
            /* rem follows the html size, means it will be 3 X 10 = 30px */
            font-size: 3rem;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1 id="first">This is paragraph</h1>
        <h1 id="second">This is second paragraph</h1>
        <h1 id="third">This is third paragraph</h1>
    </div>
</body>
</html>
```

Output:



This is paragraph

This is second paragraph

This is third paragraph

MEDIA QUERIES

What Is Media Query?

- Media queries are used when we want to customize our website's presentation according to the user's screen size. With the help of media queries, you can display different markups based upon the device's general type(mobile, desktop, tablet).
- A media query is a logical operation. Whenever a media query becomes true, then the related CSS is applied to the target element.

Syntax Of Media Query :

A media query is composed of two things: media type and expression. A media query can contain one or more expressions.

Syntax :

```
@media media-type and (media-feature)
{
```

```
/* CSS Rules to be applies*/
}
```

Example:

```
@media screen and (max-width: 800px)
{
#contents{width:90%}
}
```

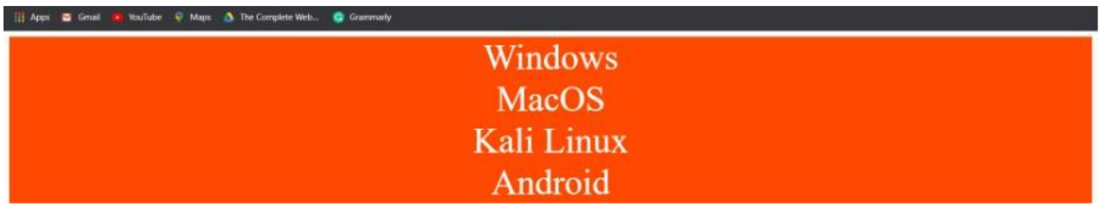
Let's understand this example :

- @media: A media query always starts with @media.
- Screen: It is the applicable media type.
- max-width: It is the media feature.
- #contents{width:90%} : It is the CSS to be applied when the conditions are met.
- Now, we will spend some time understanding how to use media queries on a website.

First of all, we have used the below CSS to design four divs.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Media Query</title>
    <style>
        .box {
            text-align: center;
            background-color: rgba(255, 196, 0, 0.959);
            color: white;
            font-size: 50px;
        }
 </style>
</head>
<body>
    <div class="box" id="box1"> Windows</div>
    <div class="box" id="box2"> MacOS</div>
    <div class="box" id="box3"> Kali Linux</div>
    <div class="box" id="box4">Android</div>
</body>
</html>
```

Output:



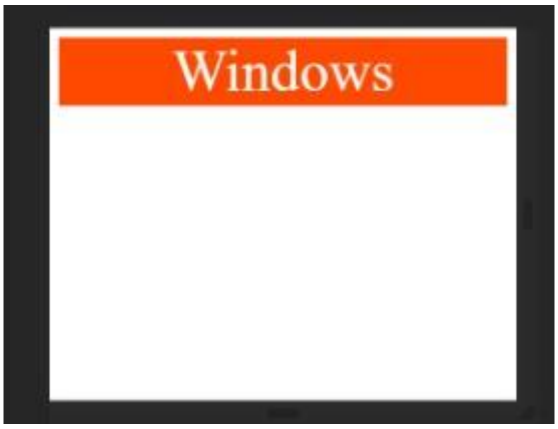
```
Windows
MacOS
Kali Linux
Android
```

Now, we are changing the display value to display:none; and then we will apply the media queries. Here is the CSS :

```
<style>
        .box {
            text-align: center;
            background-color: rgba(255, 196, 0, 0.959);
            color: white;
            font-size: 50px;
             display: none;
        }
</style>
```

In the above code, we have applied the media query to id=box1. So, whenever the page's width is 400px or more than 400px, then the display value of the box1 will be changed from display:none; to display:block; and applied CSS will be visible to the user.

Output:

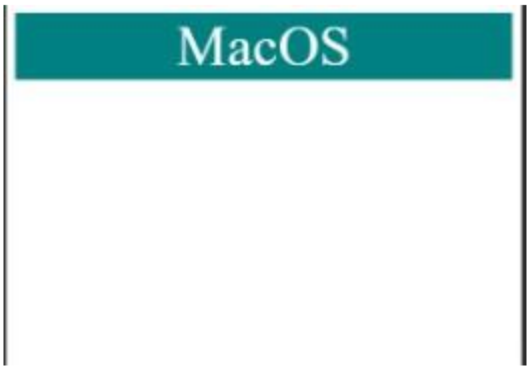


In the above image, I have used the chrome developer tools to change the page's width. You can see that the media query is applied when the page's width is set to 400px. So whenever the page width is 400px or more than 400px, then the media query for the box1 will be applied because we have set the minimum width to 400px. Similarly, we have applied a media query for the box2. Here is the CSS used:

```
/* @media (min-width: 400px){
            #box1{
                display: block;
            }
        } */
        @media (min-width: 450px) and (max-width: 600px){
            #box2{
                display: block;
                background-color: teal;
            }
        }
```

In the above code, we have set minimum width to 450px and maximum width to 600px. So the media query for the box2 will be applied whenever the screen's width is between 450px and 600px.

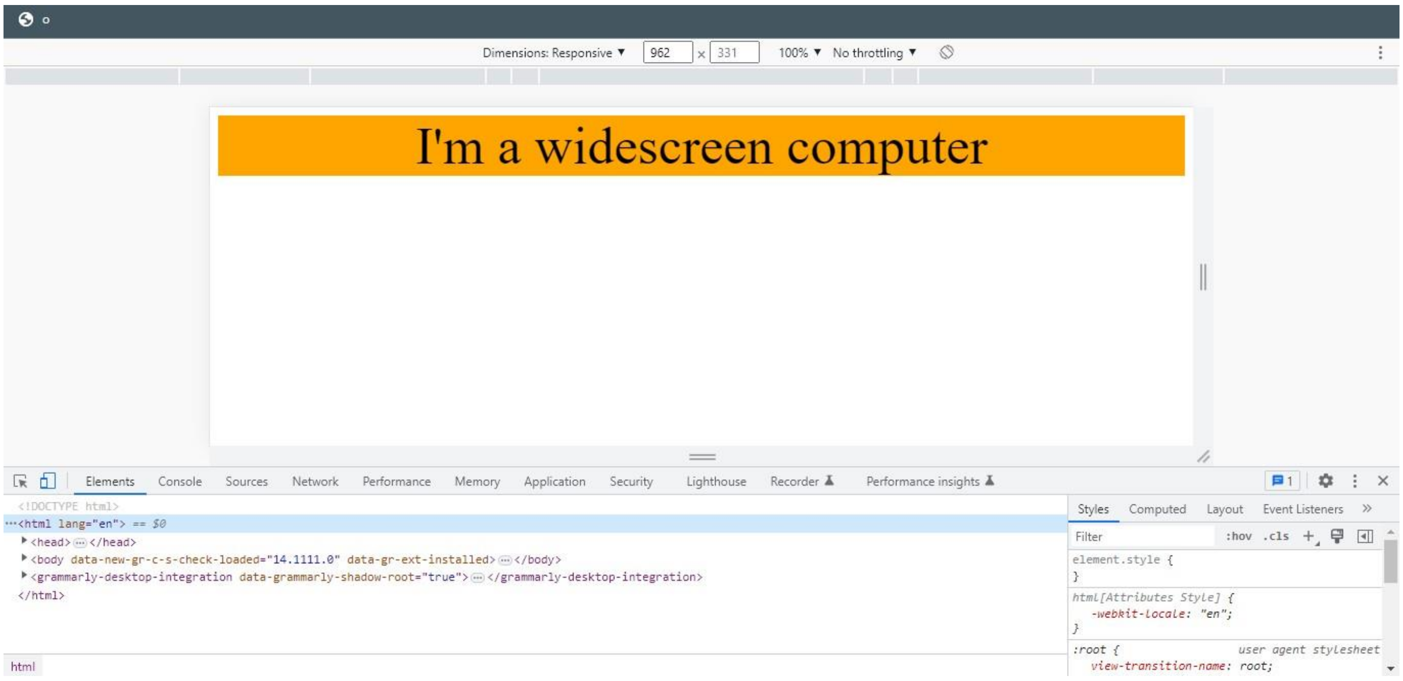
Output:



CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Media Queries</title>
    <style>
        .box{
            font-size: 52px;
            text-align: center;
            background-color: red;
            color: white;
            display: none;
        }
        @media only screen and (max-width:300px){
            #box-1{
                display: block;
                background-color: red;
            }
        }
        @media (min-width:300px) and (max-width:500px){
            #box-2{
                display: block;
                background-color: green;
                color: black;
            }
        }
        @media (min-width:500px) and (max-width:800px){
            #box-3{
                display: block;
                background-color: yellow;
                color: orange;
            }
        }
        @media (min-width:800px){
        #box-4{
            display: block;
            background-color: orange;
            color: black;
        }
        }
    </style>
</head>
<body>
    <div class="box" id="box-1">I'm a phone</div>
    <div class="box" id="box-2">I'm a tablet</div>
    <div class="box" id="box-3">I'm a computer</div>
    <div class="box" id="box-4">I'm a widescreen computer</div>
</body>
</html>
```

Output:

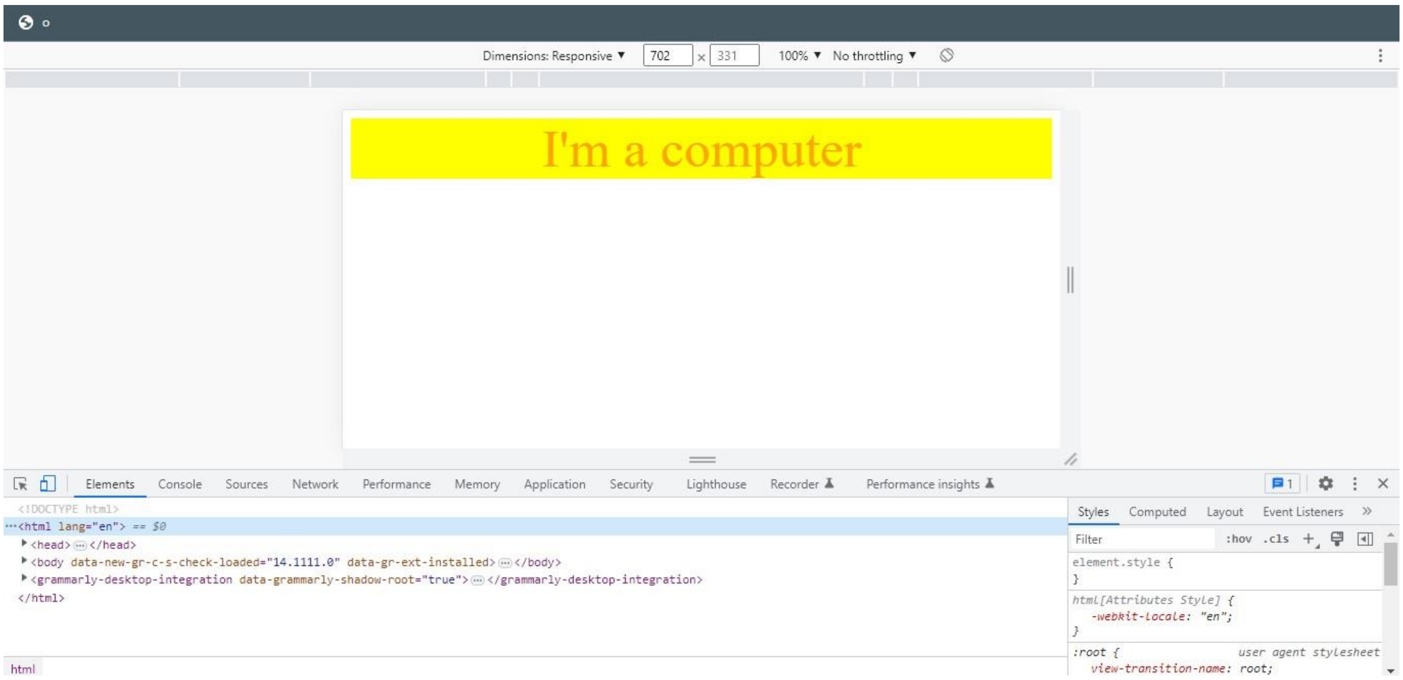


Dimensions: Responsive ▼ 962 × 331 100% ▼ No throttling ▼

I'm a widescreen computer

```
<!DOCTYPE html>
···<html lang="en"> == $0
  ▶<head>⋯</head>
  ▶<body data-new-gr-c-s-check-loaded="14.1111.0" data-gr-ext-installed>⋯</body>
  ▶<grammarly-desktop-integration data-grammarly-shadow-root="true">⋯</grammarly-desktop-integration>
</html>
```

```
element.style {
}
html[Attributes Style] {
   -webkit-locale: "en";
}
:root {                    user agent stylesheet
   view-transition-name: root;
```

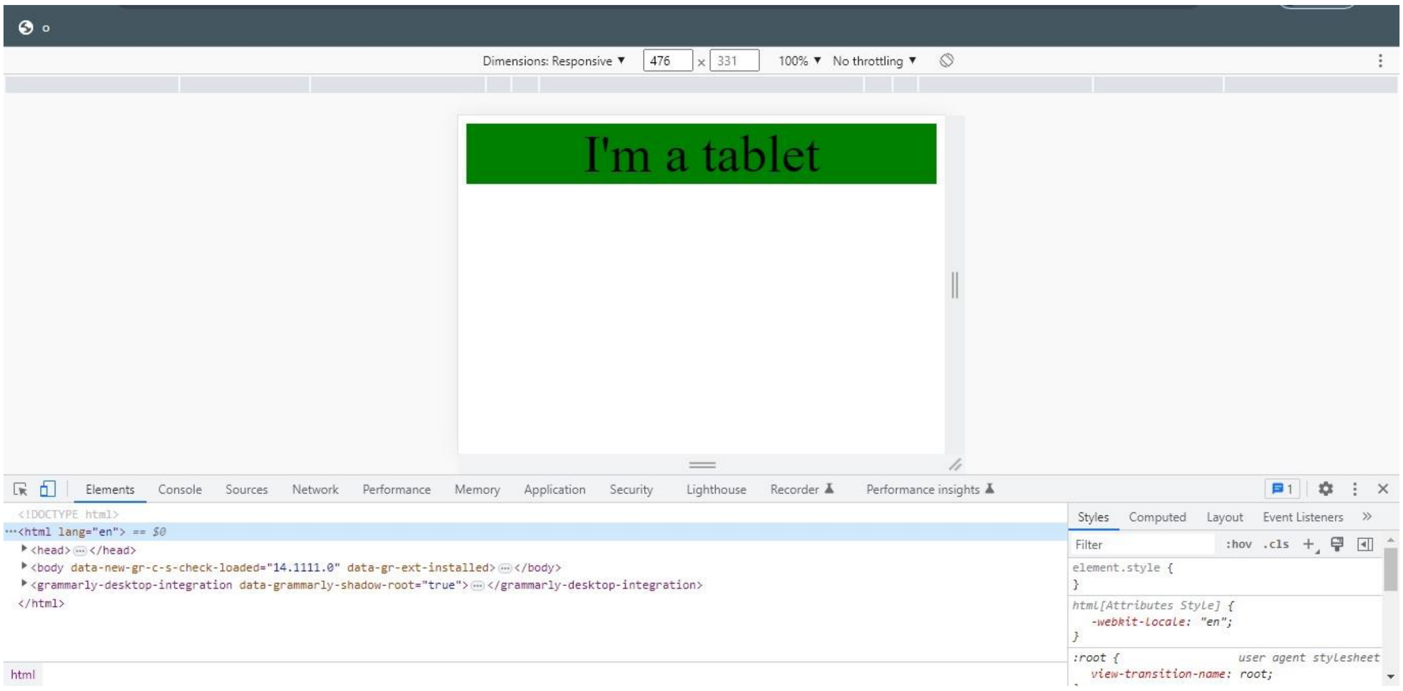


Dimensions: Responsive ▼ 702 × 331 100% ▼ No throttling ▼

I'm a computer

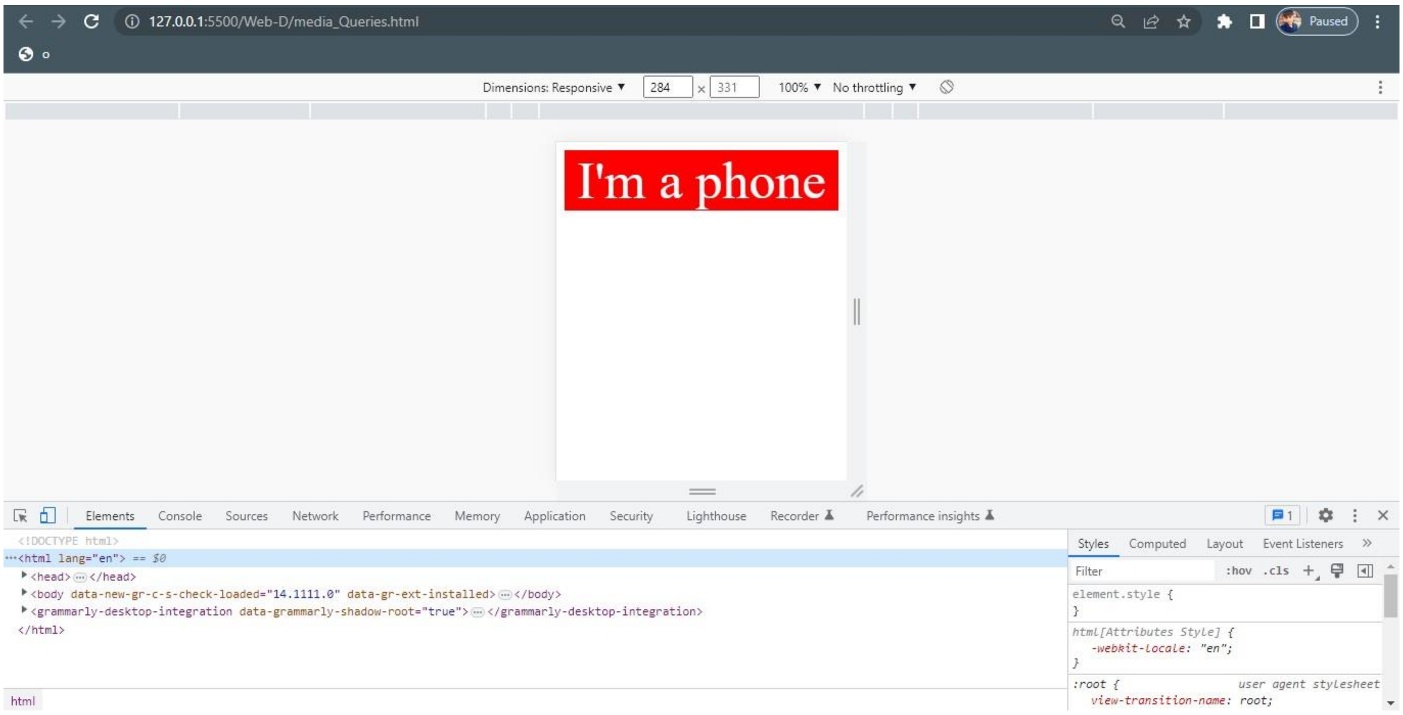
```
<!DOCTYPE html>
···<html lang="en"> == $0
  ▶<head>⋯</head>
  ▶<body data-new-gr-c-s-check-loaded="14.1111.0" data-gr-ext-installed>⋯</body>
  ▶<grammarly-desktop-integration data-grammarly-shadow-root="true">⋯</grammarly-desktop-integration>
</html>
```

```
element.style {
}
html[Attributes Style] {
   -webkit-locale: "en";
}
:root {                    user agent stylesheet
   view-transition-name: root;
```


Dimensions: Responsive ▾ 476 × 331 100% ▾ No throttling ▾
I'm a tablet
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights
Styles Computed Layout Event Listeners

```
<!DOCTYPE html>
<html lang="en"> == $0
  <head>…</head>
  <body data-new-gr-c-s-check-loaded="14.1111.0" data-gr-ext-installed>…</body>
  <grammarly-desktop-integration data-grammarly-shadow-root="true">…</grammarly-desktop-integration>
</html>
```

```
element.style {
}
html[Attributes Style] {
  -webkit-locale: "en";
}
:root {                              user agent stylesheet
  view-transition-name: root;
```


127.0.0.1:5500/Web-D/media_Queries.html
Paused
Dimensions: Responsive ▾ 284 × 331 100% ▾ No throttling ▾
I'm a phone
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights
Styles Computed Layout Event Listeners

```
<!DOCTYPE html>
<html lang="en"> == $0
  <head>…</head>
  <body data-new-gr-c-s-check-loaded="14.1111.0" data-gr-ext-installed>…</body>
  <grammarly-desktop-integration data-grammarly-shadow-root="true">…</grammarly-desktop-integration>
</html>
```

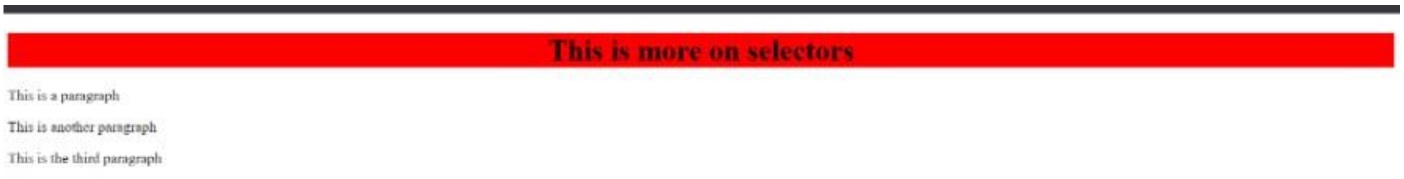
```
element.style {
}
html[Attributes Style] {
  -webkit-locale: "en";
}
:root {                              user agent stylesheet
  view-transition-name: root;
```

SELECTORS

First of all, we have used the CSS given below to design some divs and paragraph tags.

```
<style>
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Advanced CSS Selectors</title>
    <style>
        h1 {
            background-color: red;
            color: black;
            font-weight: bold;
            text-align: center;
        }
        div p {
            color: rgb(0, 0, 128);
            background-color: orange;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1> This is more on selectors</h1>
    <div class="container">
        <div class="row">
            <p> This is a paragraph</p>
        </div>
        <p> This is another paragraph</p>
    </div>
    <p> This is the third paragraph</p>
</body>
</html>
```

Output:

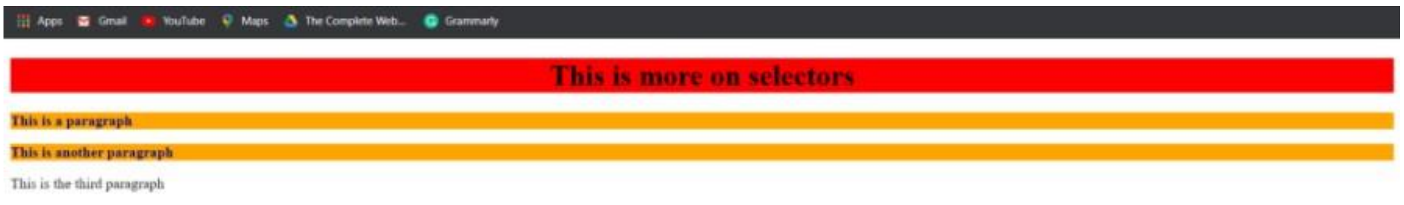


```
This is more on selectors
This is a paragraph
This is another paragraph
This is the third paragraph
```

Now, let's suppose you want to add styling to all the paragraphs inside the div. What will you do? You can use the following CSS :

```
 div p{
            color: rgb(0, 0, 128);
            background-color:orange;
            font-weight: bold;
        }
```

Here are the results :



```
This is more on selectors
This is a paragraph
This is another paragraph
This is the third paragraph
```

In the above image, you can see that the CSS is applied to the two paragraphs, but it is not applied to the third paragraph. Why? Let me answer this simple question for you. The CSS is not applied to the third paragraph because we have applied CSS only on the <p> tags inside div, and the third paragraph is not inside a div element.

This was very simple. Now let's move on to the next situation. What will you do if you want to target the <p> tags, which are the direct child of the div? So, whenever we want to target a direct child element, we use the following syntax :

```
     element> element;
```

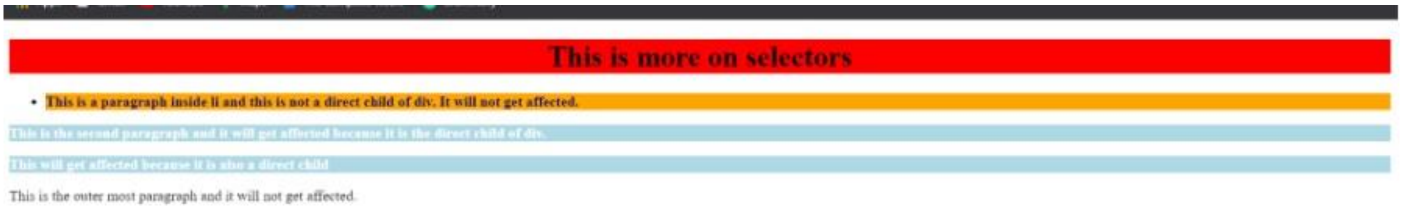
example

```
  div>p
```

In the above example, the styling will be applied to all <p> elements which are the direct child of any div element. Let's understand this with the help of paragraph tags that we created at the starting of this tutorial. Here is the CSS used :

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Advanced CSS Selectors</title>
    <style>
        h1 {
            background-color: red;
            color: black;
            font-weight: bold;
            text-align: center;
        }
        div p {
            color: rgb(0, 0, 128);
            background-color: orange;
            font-weight: bold;
        }
        <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Advanced CSS Selectors</title>
    <style>
        h1 {
            background-color: red;
            color: black;
            font-weight: bold;
            text-align: center;
        }
        div p {
            color: rgb(0, 0, 128);
            background-color: orange;
            font-weight: bold;
        }
        div>p{
            background-color: lightblue;
            color:white;
        }
    </style>
</head>
<body>
    <h1> This is more on selectors</h1>
    <div class="container">
        <div class="row">
            <ul>
                <li class="item">
                    <p>This is a paragraph inside li and this is not a direct child of
div. It will not get affected.</p>
                </li>
            </ul>
            <p> This is the second paragraph and it will get affected because it is the
direct child of div.</p>
        </div>
        <p> This is the third pargraph and it will get affected because it is also a
direct child</p>
    </div>
    <p> This is the outer most paragraph and it will not get affected. </p>
</body>
</html>
```

Output:



This is more on selectors

- This is a paragraph inside li and this is not a direct child of div. It will not get affected.

This is the second paragraph and it will get affected because it is the direct child of div.

This will get affected because it is also a direct child

This is the outer most paragraph and it will not get affected.

In the above image, you can see that the CSS is applied to the second and third paragraphs because they are the direct child of div. Paragraph inside <li> is the direct child of <li> and not of <div>. Also, in the case of the outermost paragraph, the parent element is the body and not div; therefore, no styling is applied to it. Now, we will talk about one more type of CSS selectors, i.e., div+p.

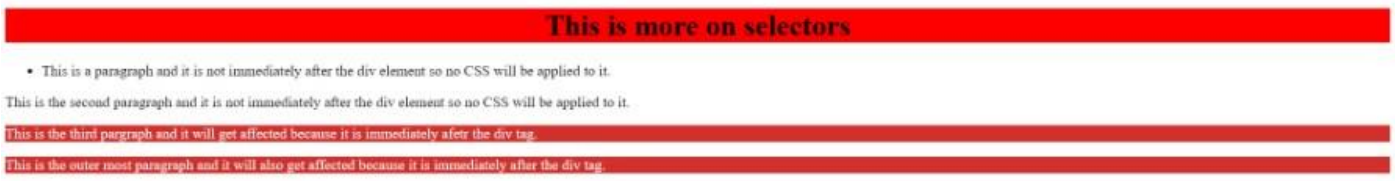
div+p :

There might be situations where you want to select the <p> tags that are immediately after the <div> elements. In such cases, we use the div+p selectors. Let's understand this with the help of example given below:

Html and CSS used:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Advanced CSS Selectors</title>
    <style>
        h1 {
            background-color: red;
            color: black;
            font-weight: bold;
            text-align: center;
        }
        /* div p {
            color: rgb(0, 0, 128);
            background-color: orange;
            font-weight: bold;
        } */
        /* div>p{
            background-color: lightblue;
            color:white;
        } */
      div+p{
          color: white;
          background-color:#D2302C;
      }
    </style>
</head>
<body>
    <h1> This is more on selectors</h1>
    <div class="container">
        <div class="row">
            <ul>
                <li class="item">
                    <p>This is a paragraph and it is not immediately after the div element
so no CSS will be applied to it.</p>
                </li>
            </ul>
            <p> This is the second paragraph and it is not immediately after the div
element so no CSS will be applied to it.</p>
        </div>
        <p> This is the third pargraph and it will get affected because it is immediately
afetr the div tag.</p>
    </div>
    <p> This is the outer most paragraph and it will also get affected because it is
immediately after the div tag. </p>
</body>
</html>
```

Output:

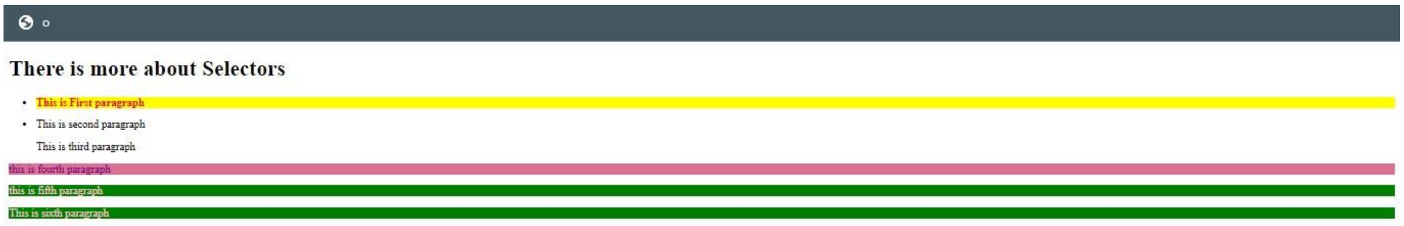


```
This is more on selectors

• This is a paragraph and it is not immediately after the div element so no CSS will be applied to it.

This is the second paragraph and it is not immediately after the div element so no CSS will be applied to it.

This is the third pargraph and it will get affected because it is immediately afetr the div tag.

This is the outer most paragraph and it will also get affected because it is immediately after the div tag.
```

CODE:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MORE ABOUT SELECTORS</title>
</head>
<style>
    /* if p is contained by any li and li is contained by div  */
    div li p{
        color: red;
        background-color: yellow;
        font-weight: bold;
    }
    /* in this if p is contained directly by div  */
    div > p{
        color: purple;
        background-color: palevioletred;
    }
    /* in this if there is p before div  */
    div + p{
        color: pink;
        background-color: green;
    }
</style>
<body>
    <h1>There is more about Selectors</h1>
    <div class="container">
        <div class="row">
            <ul>
                <li class="item"><p>This is First paragraph</p></li>
                <li>This is second paragraph</li>
                <p>This is third paragraph</p>
            </ul>
            <p>this is fourth paragraph</p>
        </div>
        <p>this is fifth paragraph</p>
    </div>
    <p>This is sixth paragraph</p>
</body>
</html>
```

Output:



There is more about Selectors

- This is First paragraph
- This is second paragraph

  This is third paragraph

this is fourth paragraph

this is fifth paragraph

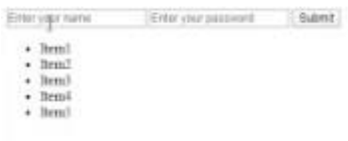
This is sixth paragraph

Attribute & nth child pseudo Selectors

We will create a very basic form by writing the HTML code as below-

```
<body>
    <div class="container">
        <h1><a href="http://google.com" target="_blank">Go to google</a></h1>
        <h1><a href="http://facebook.com" target="_self">Go to Facebook</a></h1>
        <form action="" class="form-control">
            <input type="text" placeholder="Enter your name">
            <input type="email" placeholder="Enter your email">
            <input type="password" placeholder="Enter your password">
            <input type="submit" value="Submit">
        </form>
        <ul>
            <li class="item" id="item-1">Item1</li>
            <li class="item" id="item-2">Item2</li>
            <li class="item" id="item-3">Item3</li>
            <li class="item" id="item-4">Item4</li>
            <li class="item" id="item-5">Item5</li>
            <li class="item" id="item-6">Item6</li>
            <li class="item" id="item-7">Item7</li>
            <li class="item" id="item-8">Item8</li>
            <li class="item" id="item-9">Item9</li>
            <li class="item" id="item-10">Item10</li>
        </ul>
    </div>
</body>
```

The form will look very simple as follows :



- Item1
- Item2
- Item3
- Item4
- Item5

We will now begin our styling part. Initial, we wiil set the display of the input as block-

```
 input {
            display: block;
        }
```

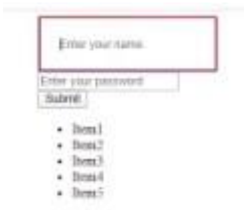
We will then try to move the container to the center of the webpage by the following code.

```
.container {
            display: block;
            width: 233px;
            margin: auto;
        }
```

To select the input of type text only, we will write the following code-

```
input[type='text'] {
            padding: 23px;
            border: 2px solid red;
        }
```

By the above code, the changes will be made only in the input form having type text.



With the help of pseudo selectors, we can target different properties by their attributes. In the same way, if we want to target our email tab, then we can write the code as follows and can apply some property to it-

```
input[type='email'] {
            color: yellow;
            border: 4px solid black;
        }
```

We will notice the black border with several other properties applied to it. In the same way, we can target more properties or tags.

Now let us see how to use nth child pseudo selectors. Suppose, if we write-

```
li:nth-child(3){
      color: cyan;
}
```

The above code will convert the text color to the cyan of the third only. What if we want to change the text color to red of every third element of the list. In that case, we can write-

```
li:nth-child(3n+3) {
            color: red;
        }
```

This will convert the text color to red of every third element in the fist.

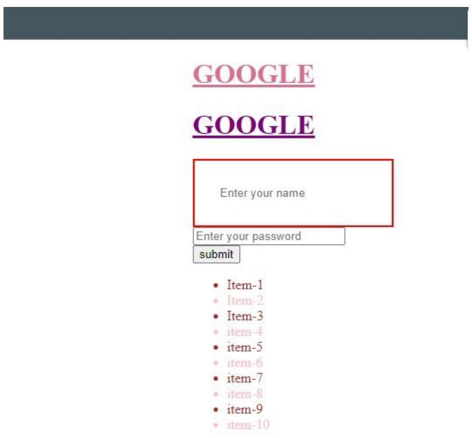
It works on the basic formula where it will convert all the items depending on the values of n starting from 0. In the same way, we can select all the even items on the list and can apply some CSS to it.

```
li:nth-child(even) {
            background-color: yellow;
        }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Attribute & nth child pseudo selector</title>
    <style>
        .container{
            display: block;
            width: 233px;
            margin: auto;
        }
        input{
            display: block;
        }
        /* used for style the input which is text type  */
        input[type='text']{
            padding:30px;
            border: 2px solid red;
        }
        /* used for style the anchor tag which has a target  */
        a[target]{
            color: palevioletred;
        }
        /* used for style the anchor tag which has a target='_self' */
        a[target='_self']{
            color: purple;
        }
        /* it is used to style the 3rd li  */
        li:nth-child(3){
            color: red;
        }
        /* 2n+0 will change the color of every second li child  */
        li:nth-child(2n+0){
            color: purple;
        }
        /* It will change color of every even no of li  */
        li:nth-child(even){
            color: pink;
        }
        /* It will change the color of every odd no of li  */
        li:nth-child(odd){
            color: brown;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1><a href="http://google.com" target="_blank">GOOGLE</a></h1>
        <h1><a href="http://google.com" target="_self">GOOGLE</a></h1>
        <form action="" class="form-control">
            <input type="text" name="" id="" placeholder="Enter your name">
            <input type="password" name="" id="" placeholder="Enter your password">
            <input type="submit" value="submit">
        </form>
        <ul>
            <li class="item" id="item-1">Item-1</li>
            <li class="item" id="item-2">Item-2</li>
            <li class="item" id="item-3">Item-3</li>
            <li class="item" id="item-4">item-4</li>
            <li class="item" id="item-5">item-5</li>
            <li class="item" id="item-6">item-6</li>
            <li class="item" id="item-7">item-7</li>
            <li class="item" id="item-8">item-8</li>
            <li class="item" id="item-9">item-9</li>
            <li class="item" id="item-10">item-10</li>
        </ul>
    </div>
</body>
</html>
```

Output:



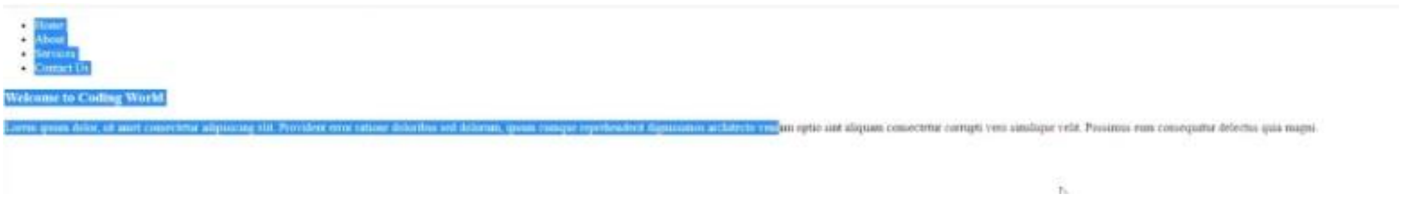
```
GOOGLE

GOOGLE

Enter your name
Enter your password
submit

• Item-1
• Item-2
• Item-3
• item-4
• item-5
• item-6
• item-7
• item-8
• item-9
• item-10
```

Before and After Pseudo Selectors

Start by writing the basic HTML code as-

```
<body>
    <header>
        <nav class="navbar">
            <ul class="navigation">
                <li class="item">Home</li>
                <li class="item">About</li>
                <li class="item">Services</li>
                <li class="item">Contact Us</li>
            </ul>
        </nav>
    </header>
        <section>
            <h1> Welcome to Coding World</h1>
            <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Provident error
ratione doloribus sed dolorum,
                ipsum cumque reprehenderit dignissimos architecto veniam optio sint
aliquam consectetur corrupti vero
                similique velit. Possimus eum consequatur delectus quia magni.</p>
        </section>
       </body>
```

By running the above code, we will get a very simple outlet of a website as shown below-



- Home
- About
- Services
- Contact Us

Welcome to Coding World

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Provident error ratione doloribus sed dolorum, ipsum cumque reprehenderit dignissimos architecto veniam optio sint aliquam consectetur corrupti vero similique velit. Possimus eum consequatur delectus quia magni.

We will then add some styling in the body tag of the website-

```
 body {
           margin: 0;
           padding: 0;
           background-color: black;
           color: white;
        }
```

To make the contents appear in a single line, we can try by setting the display property as flex.

```
.navigation {
          font-family: 'Bree Serif', serif;
           font-size: 20px;
           display: flex;
        }
```

We can then style our list items and add some padding to it-

```
 li {
            list-style: none;
            padding: 20px 23px;
        }
```

Now let us discuss what are before and after pseudo selectors. The :: before pseudo-element can be used to insert some content before the content of an element. The ::after pseudo-element can be used to insert some content after the content of an element.

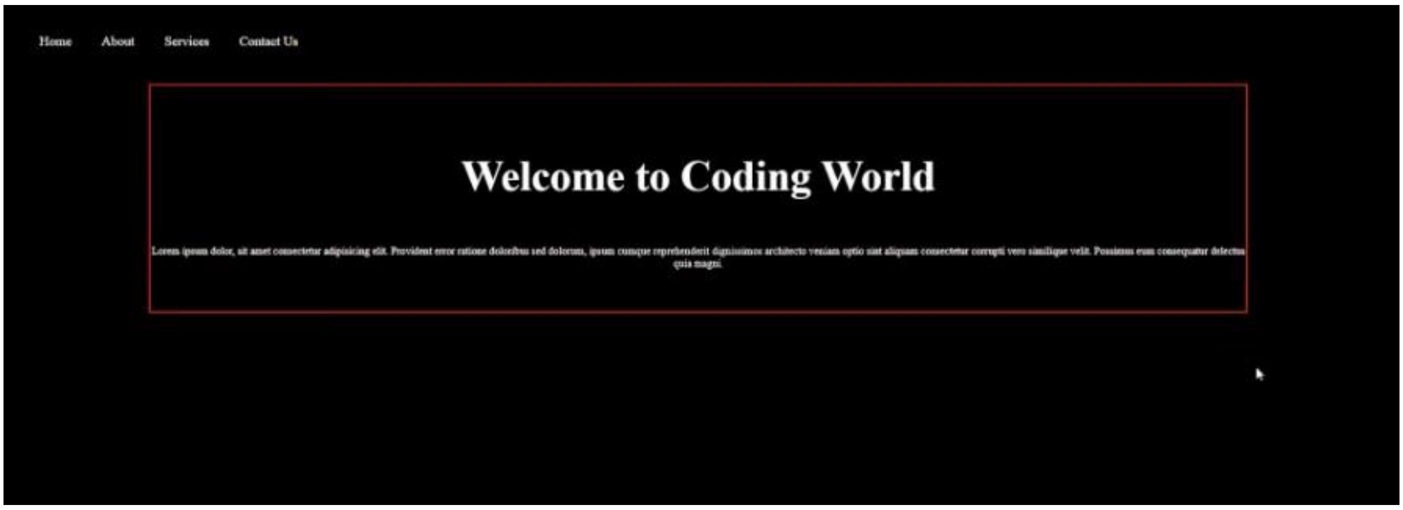
We can style the section tag in our HTML with the help of flexbox property and also make some other changes as follows-

```
section {
            height: 344px;
            font-family: 'Bree Serif', serif;
            margin: 3px 230px;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
        }
```

Then increase the font size of the heading by 4 rems and take the content of the paragraph in the center as follows-

```
 h1 {
            font-size: 4rem;
        }

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

So our page will look as follows-



Welcome to Coding World

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Provident error ratione doloribus sed dolorum, ipsum cumque reprehenderit dignissimos architecto veniam optio sint aliquam consectetur corrupti vero similique velit. Possimus eum consequatur delectus quia magni.

We can also select the fonts of our choice from Google Fonts. Now place the background image on the website. You can select different background images **from Unsplash.** You will be able to generate random background images from here. You can place the URL of the image in the **<body>** tag while styling. But we will notice that the image generated on our website does cover the whole page and make it difficult in reading the text.

To adjust this problem, we can add content before the header. In that content, we can add a background image, makes its position as absolute, and set its width and height as follows-

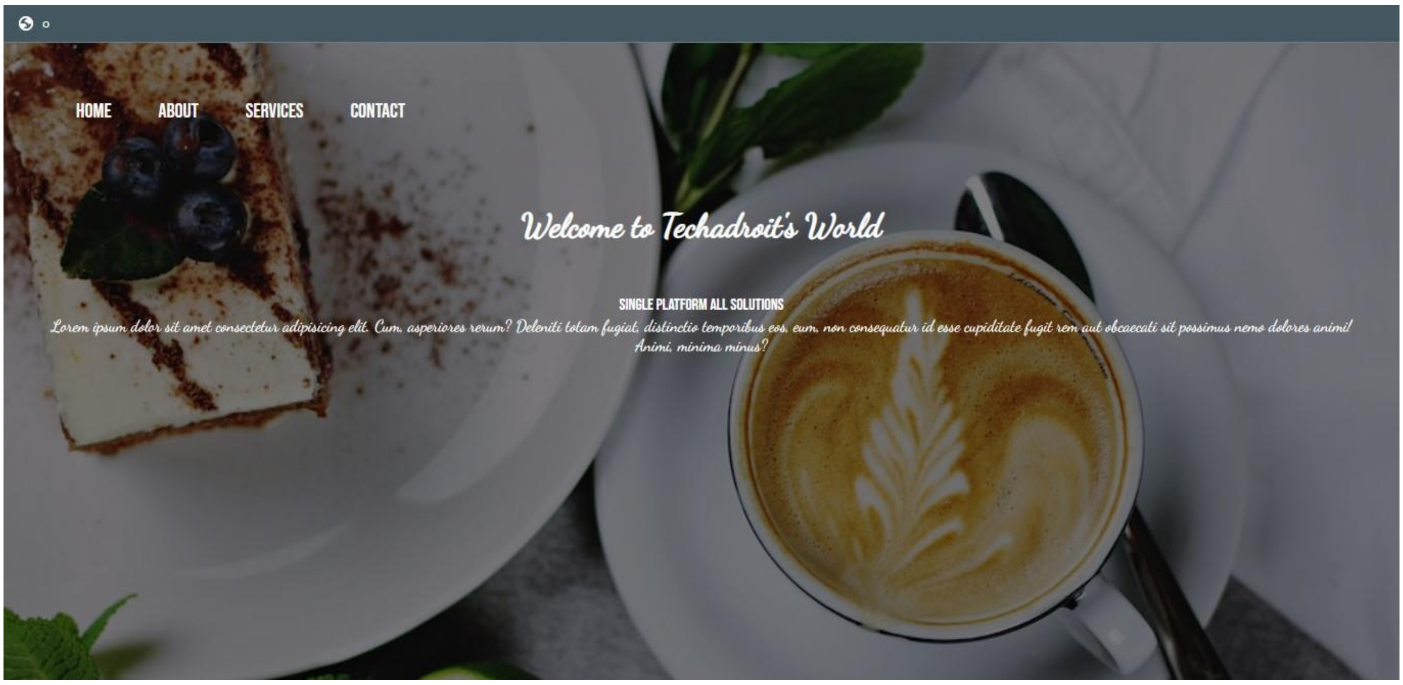
```
 header::before{
           background: url('https://source.unsplash.com/collection/190727/1600x900') no-
repeat center center/cover;
           content: "";
           position: absolute;
           top:0;
           left: 0;
           width: 100%;
           height: 100%;
           z-index: -1;
           opacity: 0.3;
       }
```

z-index property used here is for making the contents appear above the background image. We can make the image light by changing its opacity.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Before and After pseudo selector</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link
href="https://fonts.googleapis.com/css2?family=Bebas+Neue&family=Dancing+Script&family=Ubu
ntu:ital,wght@1,300&display=swap" rel="stylesheet">
    <style>
        body{
            /* height: 744px; */

            font-family: 'Bebas Neue', sans-serif;
            background-color: black;
            color: white;
        }
        header::before{
            /* margin: 0px;
            padding: 0px;
            height: 100vh; */
            background: url(https://source.unsplash.com/collection/190727/1600X900);
            background-size: cover;
            background-repeat: no-repeat;
            content:"";
            position:absolute;
            top: 0px;
            left: 0px;
            width: 100%;
            height: 100%;
            z-index: -1;
            opacity: 0.5;
        }
        .navigation{
            font-size: 20px;
            display: flex;
        }
        li{
            list-style: none;
            padding: 34px 23px;
        }
        section{
            font-family: 'Dancing Script', cursive;
            height: 180px;
            /* border: 2px solid red; */
            margin: 3px 23px;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
        }
        /* section::before will add the content before section   */
        /* section::before{
            content:"I'm not a robot"
        } */
        /* this will add the content after section   */
        /* section::after{
            content: "Yes, I'm a robot"
        } */
        section p::before{
            font-family: 'Bebas Neue', sans-serif;
            content: "Single Platform all Solutions"
        }
        h3{
            font-size: 2rem;
        }
        p{
            text-align: center;
        }
    </style>
</head>
<body>
    <header>
        <nav class="navbar">
            <ul class="navigation">
                <li class="item-1">Home</li>
                <li class="item-2">About</li>
                <li class="item-3">Services</li>
                <li class="item-4">Contact</li>
            </ul>
        </nav>
    </header>
        <section>
            <h3>Welcome to Techadroit's World</h3>
            <p><br>Lorem ipsum dolor sit amet consectetur adipisicing elit. Cum,
asperiores rerum? Deleniti totam fugiat, distinctio temporibus eos, eum, non consequatur
id esse cupiditate fugit rem aut obcaecati sit possimus nemo dolores animi! Animi, minima
minus?</p>
        </section>
    <!-- <hr> -->
</body>
</html>
```

Output:



Welcome to Techadroit's World

SINGLE PLATFORM ALL SOLUTIONS

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cum. asperiores rerum? Deleniti totam fugiat. distinctio temporibus eos. eum. non consequatur id esse cupiditate fugit rem aut obcaecati sit possimus nemo dolores animi! Animi. minima minus?

Box Shadow and Text Shadow

The **box-shadow CSS** property adds shadow effects around an element's frame. We can set multiple effects separated by commas. It is described by X and Y offsets relative to the element, blur and spread radius, and colour. The text-shadow CSS property adds a shadow to text. It accepts a comma-separated list of shadows to be applied to the text and any of its decorations. It is also described by some combination of X and Y offsets from the element, blur radius, and colour.

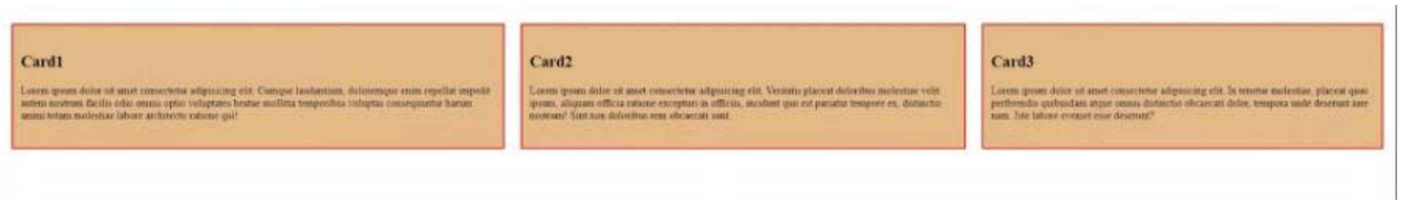
We will start by writing our HTML code. Here we will make three **divs** with the class as a card. So the code will be as follows-

```
<body>
    <div class="container">
        <div class="card" id="card-1">
           <h2>This is C++ Course</h2>
           <p>I have started C++ course which does not mean that we will stop this course. We will continue this course to completion. Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque laudantium, doloremque enim repellat impedit autem nostrum facilis odio omnis optio voluptates beatae mollitia temporibus voluptas consequuntur harum animi totam molestiae labore architecto ratione qui!</p>
        </div>
        <div class="card" id="card-2">
            <h2>This is HTML Course</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis placeat doloribus molestiae velit ipsum, aliquam officia ratione excepturi in officiis, incidunt quo est pariatur tempore ex, distinctio nostrum! Sint non doloribus rem obcaecati sunt.</p>
        </div>
        <div class="card" id="card-3">
           <h2>Card3</h2>
           <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. In tenetur molestiae, placeat quas perferendis quibusdam atque omnis distinctio obcaecati dolor, tempora unde deserunt iure nam. Iste labore eveniet esse deserunt?</p>
        </div>
    </div>
</body>
```

We will then add the margin, padding and background image to the class- .card as follows-

```
.card{
    padding: 23px 12px;
    margin: 23px 12px;
    background-color: burlywood;
}
```

Our page will look as follows-



Card1

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque laudantium, doloremque enim repellat impedit autem nostrum facilis odio omnis optio voluptates beatae mollitia temporibus voluptas consequuntur harum animi totam molestiae labore architecto ratione qui!

Card2

Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis placeat doloribus molestiae velit ipsum, aliquam officia ratione excepturi in officiis, incidunt quo est pariatur tempore ex, distinctio nostrum! Sint non doloribus rem obcaecati sunt.

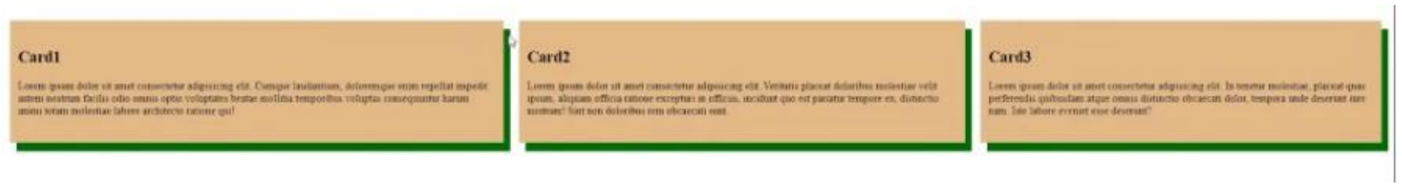
Card3

Lorem ipsum dolor sit amet consectetur adipisicing elit. In tenetur molestiae, placeat quas perferendis quibusdam atque omnis distinctio obcaecati dolor, tempora unde deserunt iure nam. Iste labore eveniet esse deserunt?

Now let us understand how to apply the box-shadow effect in the CSS. The basic syntax is as-

```
box-shadow: 10px 13px green;
```

Now the boxes inside the webpage will appear as-



Card1

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque laudantium, doloremque enim repellat impedit autem nostrum facilis odio omnis optio voluptates beatae mollitia temporibus voluptas consequuntur harum animi totam molestiae labore architecto ratione qui!

Card2

Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis placeat doloribus molestiae velit ipsum, aliquam officia ratione excepturi in officiis, incidunt quo est pariatur tempore ex, distinctio nostrum! Sint non doloribus rem obcaecati sunt.

Card3

Lorem ipsum dolor sit amet consectetur adipisicing elit. In tenetur molestiae, placeat quas perferendis quibusdam atque omnis distinctio obcaecati dolor, tempora unde deserunt iure nam. Iste labore eveniet esse deserunt?

If we write the above values in the negative, the shadow will move towards up. In the same way, we can apply the blur-radius property to the boxes. This property is used to make the border blur. The other property is spread-radius. It is used to spread the color around the box. To make all these changes inside the box, we can use the inset property as follows-

```
box-shadow: inset 3px 5px green;
```

Now let us see how to use text-shadow. For this, we will make the changes in the heading tag as follows-

```
.card h2{
      text-shadow: 3px 4px red;
}
```

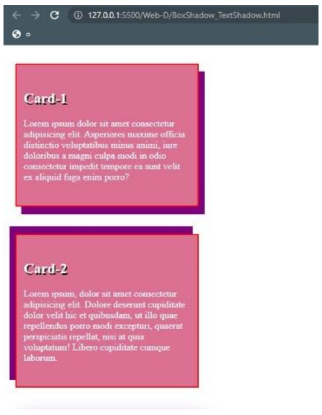
By writing this, you will see the changes as follows-

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Box SHadow and Text Shadow</title>
    <style>
        .container{
            display: flex;
            flex-direction: column;
            width: 333px;
            color: white;
            /* margin: 200px; */
            /* justify-content: space-evenly; */
        }
        .card{
            padding: 23px 12px;
            margin: 23px 12px;
            border: 2px solid red;
            background-color: palevioletred;

            /* Box shadow for top  */
            box-shadow: -10px -13px purple;
        }
        #card-1{
            /* Box shadow for bottom  */
            box-shadow: 10px 13px purple;
        }
        #card-3{
            /* Box-shadow: offset-x offset-y blur-radius  ;  */
            box-shadow: 10px 13px 30px purple;
        }
        #card-4{
            /* Box-shadow: offset-x offset-y blur-radius spread-radius ;  */
            box-shadow: 10px 13px 30px 40px purple;
        }
        #card-5{
            /* Box shadow inside the box  */
            box-shadow: inset 8px 10px purple;

        }
        #card-6{
            /* for multiple box shadows  */
            box-shadow: 10px 13px green, 15px 18px purple;

        }
        .card h2{
            text-shadow: 2px 3px black;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="card" id="card-1">
            <h2>Card-1</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores maxime
officia distinctio voluptatibus minus animi, iure doloribus a magni culpa modi in odio
consectetur impedit tempore ea sunt velit ex aliquid fuga enim porro?</p>
        </div>
        <div class="card" id="card-2">
            <h2>Card-2</h2>
            <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit. Dolore deserunt
cupiditate dolor velit hic et quibusdam, ut illo quae repellendus porro modi excepturi,
quaerat perspiciatis repellat, nisi at quia voluptatum! Libero cupiditate cumque
laborum.</p>
        </div>
        <div class="card" id="card-3">
            <h2>Card-3</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate quia eum
quos sint, earum doloribus? Facilis, beatae? Aut numquam qui ipsum? Incidunt cum sunt
voluptatibus neque, eveniet ducimus placeat mollitia necessitatibus harum reprehenderit
eos?</p>
        </div>
        <div class="card" id="card-4">
            <h2>Card-4</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate quia eum
quos sint, earum doloribus? Facilis, beatae? Aut numquam qui ipsum? Incidunt cum sunt
voluptatibus neque, eveniet ducimus placeat mollitia necessitatibus harum reprehenderit
eos?</p>
        </div>
        <div class="card" id="card-5">
            <h2>Card-5</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate quia eum
quos sint, earum doloribus? Facilis, beatae? Aut numquam qui ipsum? Incidunt cum sunt
voluptatibus neque, eveniet ducimus placeat mollitia necessitatibus harum reprehenderit
eos?</p>
        </div>
        <div class="card" id="card-6">
```

```html
            <h2>Card-6</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate quia eum quos sint, earum doloribus? Facilis, beatae? Aut numquam qui ipsum? Incidunt cum sunt voluptatibus neque, eveniet ducimus placeat mollitia necessitatibus harum reprehenderit eos?</p>
        </div>
    </div>

</body>
</html>
```

Output:



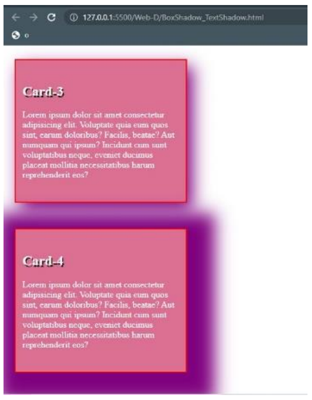
127.0.0.1:5500/Web-D/BoxShadow_TextShadow.html

Card-1

Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores maxime officia distinctio voluptatibus minus animi, iure doloribus a magni culpa modi in odio consectetur impedit tempore ea sunt velit ex aliquid fuga enim porro?

Card-2

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Dolore deserunt cupiditate dolor velit hic et quibusdam, ut illo quae repellendus porro modi excepturi, quaerat perspiciatis repellat, nisi at quia voluptatum! Libero cupiditate cumque laborum.



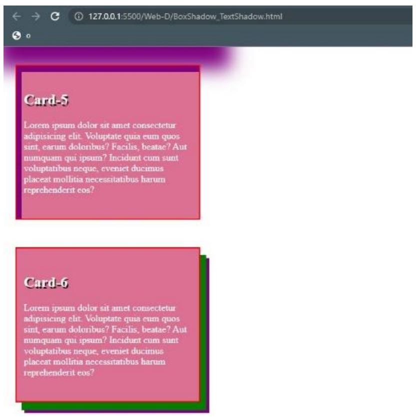
127.0.0.1:5500/Web-D/BoxShadow_TextShadow.html

Card-3

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate quia eum quos sint, earum doloribus? Facilis, beatae? Aut numquam qui ipsum? Incidunt cum sunt voluptatibus neque, eveniet ducimus placeat mollitia necessitatibus harum reprehenderit eos?

Card-4

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate quia eum quos sint, earum doloribus? Facilis, beatae? Aut numquam qui ipsum? Incidunt cum sunt voluptatibus neque, eveniet ducimus placeat mollitia necessitatibus harum reprehenderit eos?



127.0.0.1:5500/Web-D/BoxShadow_TextShadow.html

Card-5

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate quia eum quos sint, earum doloribus? Facilis, beatae? Aut numquam qui ipsum? Incidunt cum sunt voluptatibus neque, eveniet ducimus placeat mollitia necessitatibus harum reprehenderit eos?

Card-6

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate quia eum quos sint, earum doloribus? Facilis, beatae? Aut numquam qui ipsum? Incidunt cum sunt voluptatibus neque, eveniet ducimus placeat mollitia necessitatibus harum reprehenderit eos?

Variables & Custom Properties

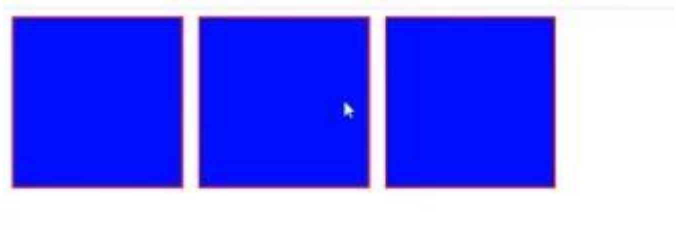
The variables in CSS are different than those of programming languages. For the HTML part, we will make a container and three boxes inside the divs as follows-

```
<body>
    <div class="container">
        <div class="box"></div>
        <div class="box"></div>
        <div class="box"></div>
    </div>
</body>
```

Let us style the box as follows-

```
.box{
     width: 200px;
     height: 200px;
     background-color: blue;
     border: 2px solid red;
     margin: 2px 9px;
}
```

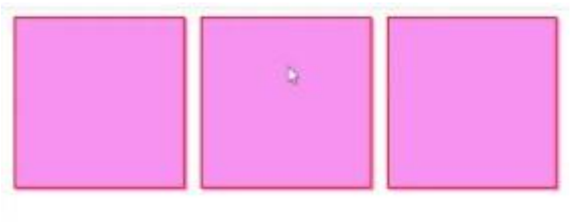
The result of the above code will be as follows-



Now we will understand the concept of variables. Suppose, we want to create a variable for the background color. We can create it by '--' symbol. Variables in CSS helps us to assign the same properties to different elements. Let us analyze it with the code given below-

```
.box{
            --box-color: violet;
            width:200px;
            height: 200px;
            background-color: var(--box-color);
            border: 2px solid var(--box-color);
            box-shadow: 3px 3px var(--box-color);
            margin: 2px 9px;
        }
```

Here, we are using the variable properties to three elements i.e. background color, border, and box-shadow. The change will be as follows-



The important point to remember about these variables is it can be used within its scope only. To make it work, we can write it again or can use the --root property. To make it understand clearer, we can make a global variable in terms of programming language. Let us understand the code below-

```
:root{
            --primary-color: blue;
            --danger-color: red;
            --maxw: 333px;
        }
```

Any custom properties written in the root variable can be accessed anywhere in the code. In most of the cases, we use primary color and danger color as shown in the above example. We have to modify the violet color with the primary color and danger color in the box class as follows-

```
   .box{
            width:200px;
            height: 200px;
            background-color: var(--primary-color);
            border: 2px solid var(--danger-color);
            box-shadow: 3px 3px var(--box-color);
            margin: 2px 9px;
        }
```

Now let us modify the container and add the maxw property in it from the root with additional some properties. The code is as follows-

```
 .container{
            max-width: var(--maxw);
            margin: auto;
            background-color: var(--danger-color);
            display: flex;
            align-items: center;
            justify-content: center;
            /* background-color: var(--box-color); */
        }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```html
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Variables and Custom Properties</title>
    <style>
        /* we can declare global variables by using ' : ' and then name and last body
{}  */
        :root{
            --primary-color:green;
            --max-w:333px;
            --hell-color: grey;
        }
        .box{
            /* we can declare local variables using ' -- ' sign  */
            --box-color:yellow;
            --shadow:red;
            width: 200px;
            height: 200px;
            background-color: var(--box-color);
            border: 2px solid orange;
            box-shadow: 3px 4px var(--shadow);
            margin: 2px 3px;
        }
        .box2{
            width: 200px;
            height: 200px;
            /* using global variable  */
            background-color: var(--primary-color);
            border: 2px solid orange;
            box-shadow: 3px 4px var(--shadow);
            margin: 2px 3px;
        }
        .container{
            max-width: var(--max-w);
            margin: auto;
            background-color: var(--hell-color);
            display: flex;
            align-items: center;
            justify-content: center;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box"></div>
        <div class="box"></div>
        <div class="box"></div>
        <div class="box2"></div>
    </div>
</body>
</html>
```

Output:

Creating Animations & Keyframes

Creating animations with CSS is the most exciting part of the whole web development.

```
<body>
    <div class="container">
        <div class="box">
            This is a box
        </div>
    </div>
</body>
```

Let us now style our container and box by adding some of the CSS as follows-

```
.container {
        background-color: greenyellow;
    }

    .box {
        background-color: green;
        width: 250px;
        height: 250px;
        position: relative;
}
```

The position of the box is set to be relative so that we can move it within our webpage.

For making our animation, we need to start by giving the animation-name. We can give any name here. It is just used to define our animation. The code for designing the animation is as follows-

```
 .box {
        background-color: green;
        width: 250px;
        height: 250px;
        position: relative;
        /* animation-name: harry1; */
        animation-name: sanjay2;
        animation-duration: 8s;
        animation-iteration-count: 1;
```

```
}
```

In the above example, we are using the animation-name as harry1. The next property used is animation-duration. It is used to decide how long our animation will run. The last property is animation-iteration-count. It is used to decide the number of times the animation will run.

Now we will define the animation we made, i.e. harry1 as follows-

```
@keyframes sanjay1 {
        from {
            width: 200px;
        }

        to {
            width: 1400px;
        }
    }
```

The **keyframes** are used to make the animation. **From and to** are used to decide how the animation will move in the webpage. In the above example, we are moving the animation harry 1 from 200px to 1400px. These types of animations are used to design scroll bars or progress bars on the webpage.

There are some other properties also to customize the animations like-

- **animation-fill-mode:**

If we want to keep the last property applied to the animation then we can set the animation-fill-property as forward as follows-

```
animation-fill-mode: forward;
```

- **animation-timing-function:**

We can define this property with three different values-

1. **ease-in**

After applying this, the animation will start slowly and becomes fast towards the end.

2. **ease-out**

After applying this, the animation will begin fastly and become slow towards the end.

3. **ease-in-out**

After applying this, the animation will start slowly, then become fast in the midway, and ends slowly.

- **animation-delay**

It is used to define the time after which the animation will start.

```
animation-delay: 3s;
```

- **animation-direction:**

This property is used to define the direction of the animation. For example, if we select it as reverse, it will move the animation in reverse direction.

```
animation-direction: reverse;
```

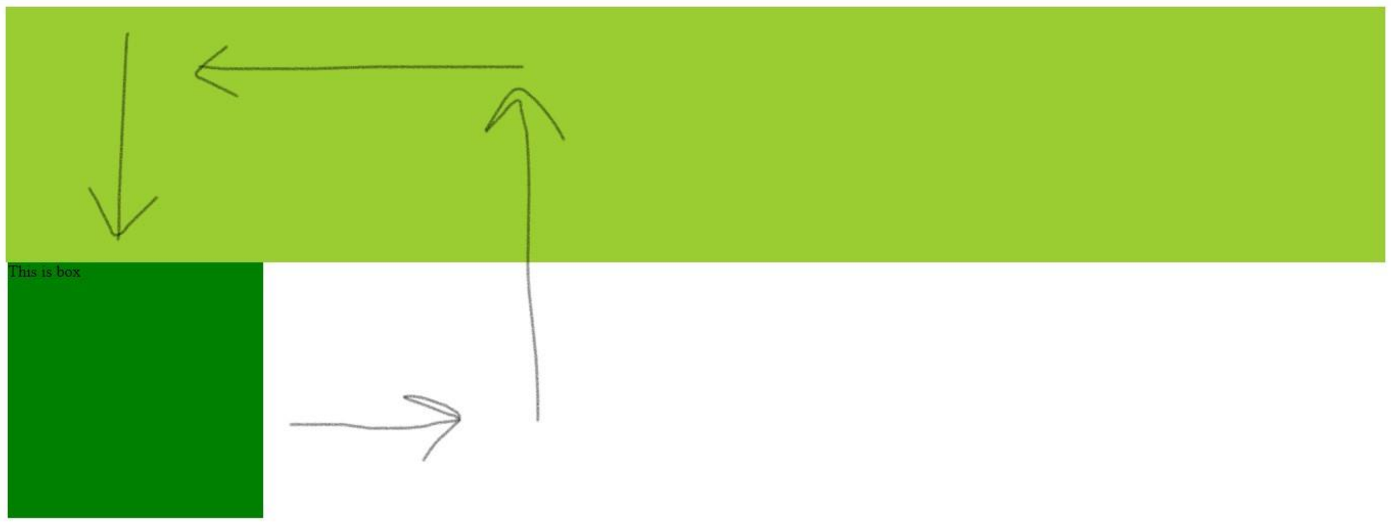
There is another method of creating animation apart from keyframes. For this, we will give the name as sanjay2.

```
  @keyframes sanjay2 {
        0%{
            top:0px;
            left:0px;
        }
        25%{
            top: 250px;
            left: 0px;
        }
        75%{
            top: 250px;
            left: 250px;
        }
        100%{
            top: 0px;
            left: 250px;
        }

    }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Animation and Keyframes</title>
    <style>
        .container{
            /* display: flex; */
            background-color: yellowgreen;
        }
        .box{
            background-color: green;
            width: 250px;
            height: 250px;
            /* we will provide the position to add animation  */
            position: relative;
            /* giving the name to the animation  */
            animation-name: sanjay-2;
            /* giving the duration(time)  */
            animation-duration: 2s;
            /* how many times animation will take place  */
            animation-iteration-count: 1;
            /* for infinity times we can use infinite  */
            /* animation-iteration-count: infinite;  */
            /* in fill mode forwards is used to stop the animation when it ends without
start to begining after ending */
            /* animation-fill-mode: forwards; */
            /* ease in is used for speed of the animation, the animation will start slowly
from start and then fast  */
            /* animation-timing-function: ease-in; */
            /* ease out is opposite of ease in, it will start fast and then slow  */
            /* animation-timing-function: ease-out; */
            /* ease-in-out is used for the starting and ending will be slow but the middle
animation will be fast  */
            /* animation-timing-function: ease-in-out; */
            /* animation delay is used to give the timing at starting point means after
how much time animation should start  */
            /* animation-delay: 3s; */
            /* animation direction is used to give the direction to the animation  */
            /* animation-direction: alternate; */

            /* The properties can be set using this shorthand  */
            /* animation: animation-name animation-duration animation-timing-function
animation-delay animation-iteration-count animation-fill-mode ;  */
            animation: sanjay-2 5s ease-in 1s 12 backwards ;
        }
        @keyframes sanjay-2{
            0%{
                top: 0px;
                left: 0px;
            }
            25%{
                top: 250px;
                left: 0px;
            }
            75%{
                top: 250px;
                left: 250px;
            }
            100%{
                top: 0px;
                left: 250px;
            }
        }
        @keyframes sanjay {
            for{
                width: 200px;
            }
            to{
                width: 900px;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box">
            This is box
        </div>
    </div>
</body>
</html>
```

Output:



Creating Transitions in CSS

CSS transitions allow us to change property values smoothly, over a given duration. It provides a way to control animation speed when changing CSS properties.

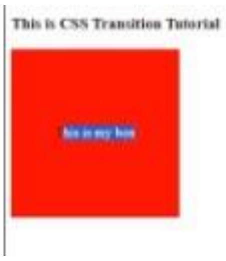
```
<body>
    <h3>This is CSS Transition Tutorial</h3>
    <div class="container">
        <div id="box">
            This is my box
        </div>
    </div>
</body>
```

Now we will add some CSS in the box to see some of the transitions effects-

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

```
#box{
        display: flex;
        height: 200px;
        width: 200px;
        background-color: red;
        justify-content: center;
        align-items: center;
}
```

The align-items as center is used here to place the text inside the box in the center as shown below.


This is CSS Transition Tutorial

Now we will make a hover effect which will change the properties when the mouse pointer will hover on the box.

```
   #box:hover{
        background-color: green;
}
```

Let us now discuss some of the transition properties-

1. **Transition-property-** It is used to decide which transition property we want to use. For example, if we want to transition background color, then we have to write-

```
transition-property: background-color;
```

2. **Transition-duration-** If we want to see the duration which is required to make the change, we can use this property. For example, if we set transition duration as 1seconds, then the transition will happen in 1 second only.

```
transition-duration: 1s;
```

3. **Transition-timing-function-** This property is used to decide the speed of transition from beginning to end. These are of three types as follows-

- **ease-in**

After applying this, the animation will start slowly and becomes fast towards the end.

- **ease-out**

After applying this, the animation will begin fastly and become slow towards the end.

- **ease-in-out**

After applying this, the animation will start slowly, then become fast in the midway, and ends slowly.

```
transition-timing-function: ease-in-out;
```

4. **Transition-delay-** It is that particular time interval after which the transition effect will start. For example, if we set it as 2s, then the transition effect will start after 2 seconds only.

```
transition-delay: 2s;
```

Also, there is one short hand property that allows us to write all these transitions in a single line. It can be written as follows-

```
transition: background-color 1s ease-in-out 2s;
```

If we want all the properties should go under transition, then we can write-

```
transition: all 1s ease-in-out 2s;
```

Now, we can add some more properties in the hover effect as follows-

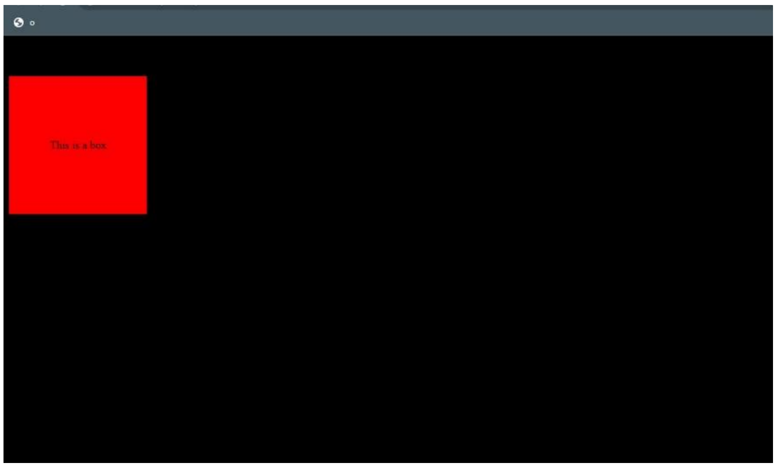
```
#box:hover{
        background-color: green;
        height: 400px;
        width: 400px;
        border-radius: 100px;
        font-size: 45px;
    }
```

CODE:

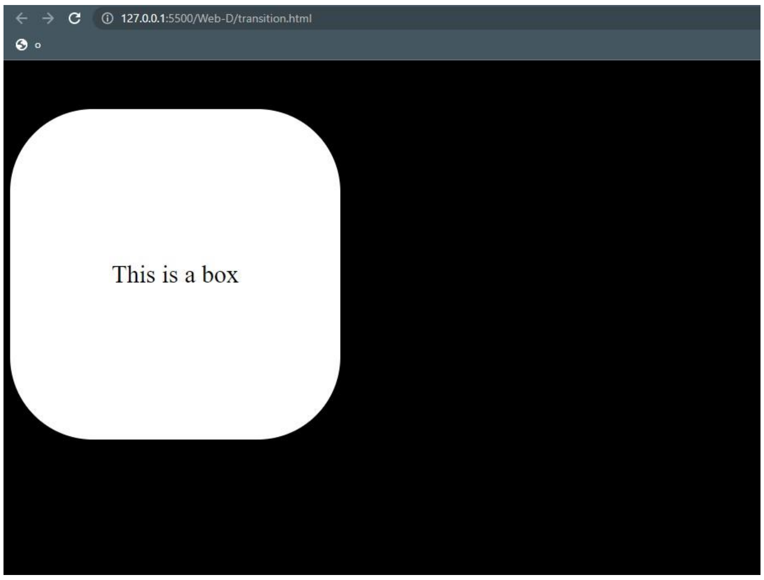
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Transitions</title>
    <style>
        body{
            background-color: black;
        }
        #box{
            display: flex;
            /* justify-content: center;
            align-content: center; */
            width: 200px;
            height: 200px;
            background-color: red;
            justify-content: center;
            align-items: center;
            /* transition-property: background-color;
            transition-duration: 1s;
            transition-timing-function: ease-in-out;
            transition-delay: 1s; */
            /* transition property in shorthand  */
            /* in this transition only background color will be transit  */
            /* transition: background-color 1s ease-in-out 1s ; */
            /* transition: all - it is used to apply transition to every element which is
present in hover  */
            transition: all 1s ease-in-out 0.3s;

        }
        #box:hover{
            background-color: white;
            height: 400px;
            width: 400px;
            border-radius: 100px;
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h3>This is transition</h3>
    <div class="container">
        <div id="box">This is a box</div>
    </div>
</body>
</html>
```

Output:

Before transition:



After transition:



Transform property in CSS

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew., elements. This property is also used in creating animations.

```
<body>
    <div class="container">
        <div class="box">This is a box</div>
    </div>
</body>
```

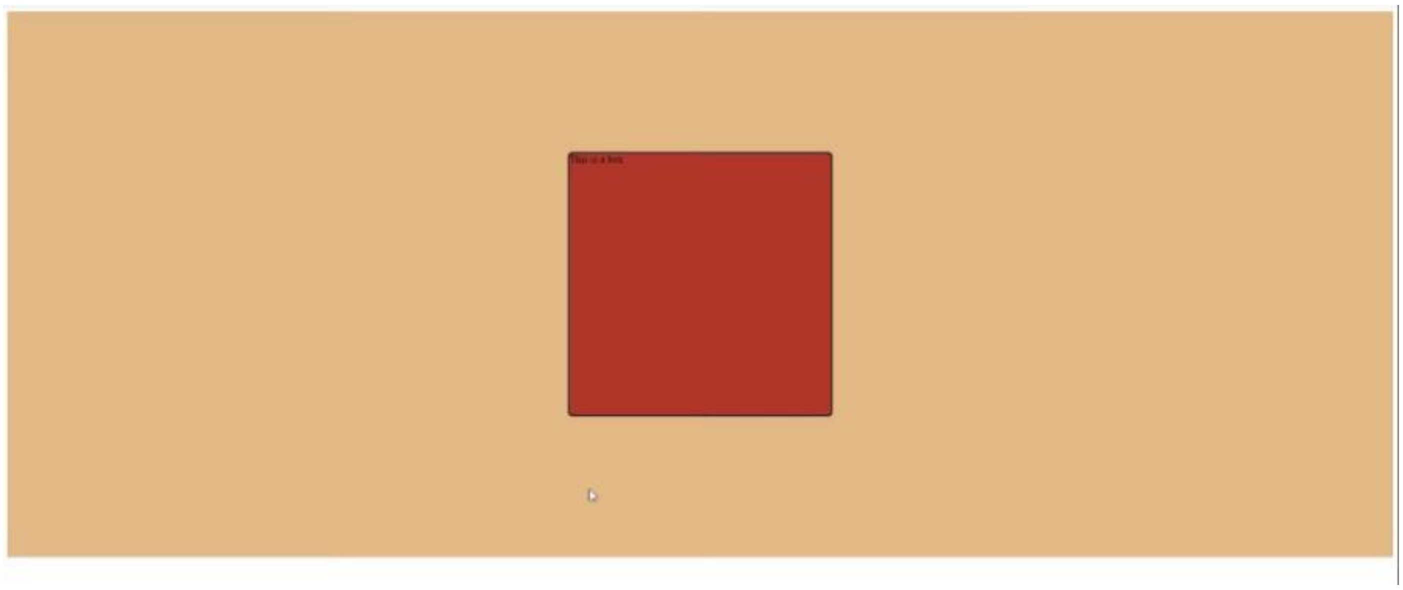
Let us now design our box with some CSS as shown below-

```
.box{
            background: brown;
            border: 2px solid black;
            border-radius: 8px;
            height: 400px;
            width: 400px;
}
```

We will also customize our container with the following code as shown below-

```
.container{
            height: 80vh;
            background-color: burlywood;
            display: flex;
            justify-content: center;
            align-items: center;
        }
```

The box and the container will look as follows-



The box will come to the center. We can reset the margin and paddings already applied as follows-

```
*{
            margin: 0px;
            padding: 0px;
        }
```

We are doing this because you must know what all obstacles occur while creating a website and how to tackle them.

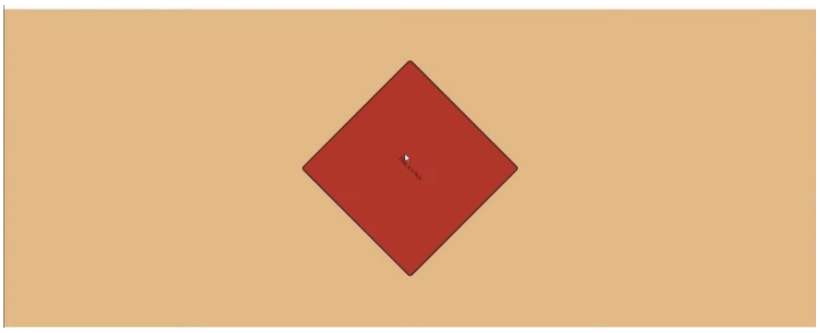
Now we can play with the box and apply different transform properties to it. However, we can add more properties to the box to make it look better.

```
.box{
            display: flex;
            align-items: center;
            justify-content: center;
}
```

We can add transform property directly to the box itself. For example, if we write

```
transform: rotate(45deg);
```

After writing the above code in the box element, it will look like-



We have to apply the transition effect in the box so that it looks like a complete transition effect. For that, we have to add the below code in the box element.-

```
transition: all 0.5s ease-in-out;
```

Now, we can ad the hover effect to add the various transition effects-

- If we want to rotate the box, we can write-

```
.box:hover{
            transform: rotate(360deg);
}
```

It will completely rotate the box to 360 degrees.

- We can also **skew** the box through certain degrees. The skew property is used sometimes when we want to put the content on one side or we want to show the 3D effect. The code is as follows-

```
.box:hover{
            transform: skew(40deg);
}
```

- We can also **scale** the box. The box will become large depending on the values we provide.

```
.box:hover{
              transform: scale(2);
}
```

In the above example, the box will grow 2 times the initial size.

- We can also **translate** or move the box in **the x or y axis** respectively by providing some values. The code for them is as follows-

```
.box:hover{
            transform: translateX(123px);
            transform: translateY(123px);
}
```

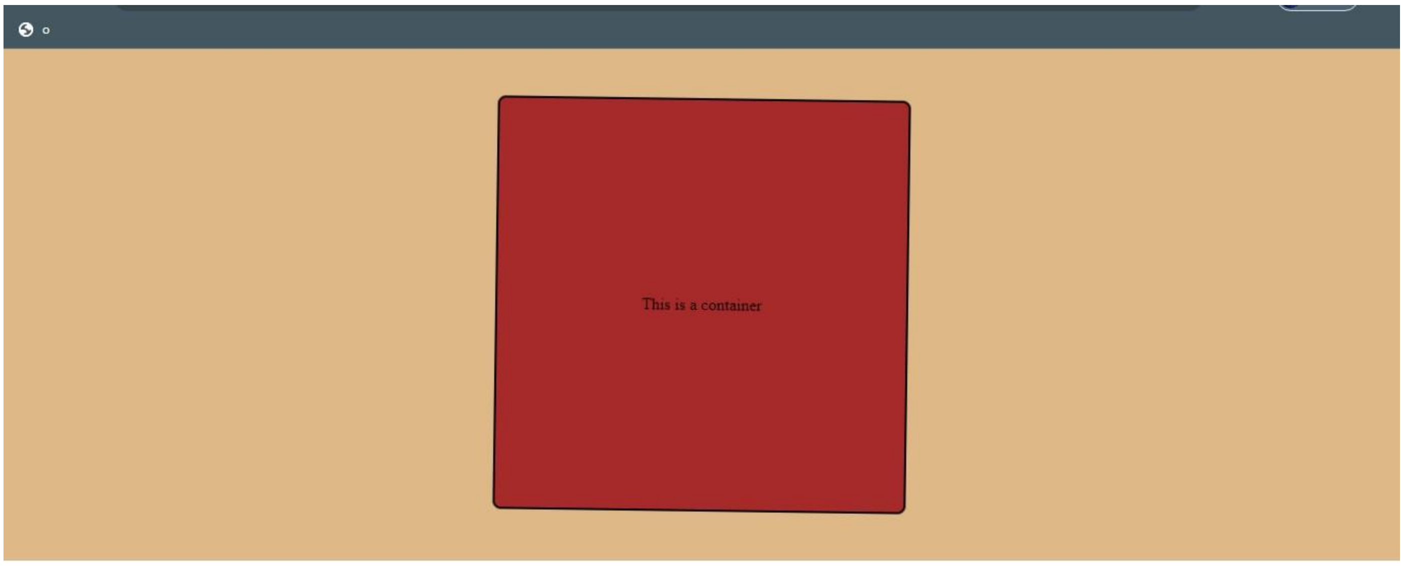
We can also write them in the same line as follows-

```
transform: translate(123px, 123px);
```

CODE:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Transform property</title>
    <style>
        *{
            /* used to give the marging and padding to all of them  */
            margin: 0px;
            padding: 0px;
        }
        .container{
            height: 80vh;
            background-color: burlywood;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        .box{
            display: flex;
            justify-content: center;
            align-items: center;
            background-color: brown;
            border-radius: 8px;
            border: 2px solid black;
            width: 400px;
            height: 400px;
            transition: all 0.5s ease-in-out;
        }
        .box:hover{
            /* this will rotate the item 360 degree  */
            transform:rotate(360deg);
            /* transform: skew(40deg); */
            /* scale will increase the size by 2  */
            /* transform: scale(2); */
            /* translateX() will change the position of an item at X-axis by 123px  */
            /* transform: translateX();
            transform: translateY(); */
            /* for both X and Y  */
            /* transform: translate(123px 123px); */
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box">This is a container</div>
    </div>
</body>
</html>
```

Output:



Introduction & Creating A Basic Grid

CSS grids are the display properties that allow us to transform any box into the grid. The main difference between flexbox and grid is, in flexbox, we can either move the box in horizontal or vertical directions but in the grid system, we can make the two-dimensional grid systems –

Let us write our HTML code to get started. We will create nine divs inside the container as follows-

```
<body>
    <div class="container">
        <div class="item">This is Item-1</div>
        <div class="item">This is Item-2</div>
        <div class="item">This is Item-3</div>
        <div class="item">This is Item-4</div>
        <div class="item">This is Item-5</div>
        <div class="item">This is Item-6</div>
        <div class="item">This is Item-7</div>
        <div class="item">This is Item-8</div>
        <div class="item">This is Item-9</div>
    </div>
</body>
```

Now we have to initialize the grid and make all its components act in two-dimensional layout. For this, we have to write-

```
.container{
        display: grid;
}
```

By writing the above code, we mean to say that all the things inside the container will now behave like a grid. The grid system should not be considered same as table because table is just a normal combination of rows and columns whereas grid is a full layout system.

Now we can define our items through CSS as follows-

```
<style>
.item{
          height: 100px;
          width: 100px;
          background-color: red;
          margin: 3px;
}
```

If we add the below code in the container as follows-

```
grid-template-columns: 300px 100px 100px;
```

But if we want that the first box should get 300px, the second 100px and the third one should be set as auto. Then we can write-

```
grid-template-columns: 300px 100px auto;
```

We can also write it using fr units as shown below-

```
grid-template-columns: 1fr 4fr 1fr;
```

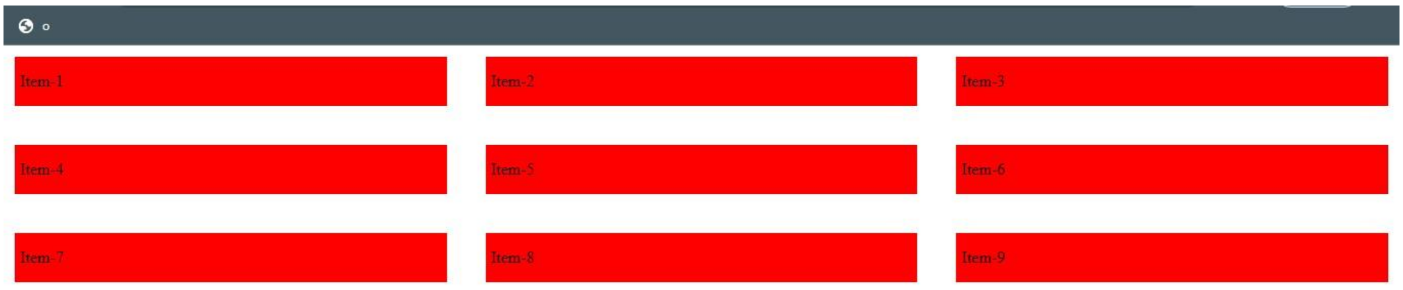
However, if there are many items to define, then we can use this code-

```
grid-template-columns: repeat(3, auto);
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS grids basic</title>
    <style>
        .container{
            display: grid;
            /* it is used to give the width in pixel to the columns */
            /* grid-template-columns: 300px 100px 100px; */
            /* grid-template-columns: 300px auto 100px; */
            /* we can also write in this form  */
            /* grid-template-columns: 1fr 4fr 1fr; */
            /* to set the no of column  */
            grid-template-columns: repeat(3, auto);
            /* to give the gap  */
            grid-gap: 2rem;
        }
        .item{
            /* height: 100px;
            width: 100px; */
            background-color: red;
            margin: 3px;
            padding: 15px 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item">Item-1</div>
        <div class="item">Item-2</div>
        <div class="item">Item-3</div>
        <div class="item">Item-4</div>
        <div class="item">Item-5</div>
        <div class="item">Item-6</div>
        <div class="item">Item-7</div>
        <div class="item">Item-8</div>
        <div class="item">Item-9</div>
    </div>
</body>
</html>
```

Output:



```
Item-1    Item-2    Item-3
Item-4    Item-5    Item-6
Item-7    Item-8    Item-9
```

Creating Rows & Gaps in Grid

We have discussed everything about grid columns and how to structure them but we have not seen anything about grid rows.

```
<body>
    <div class="grid">
        <div class="box">This is box-1</div>
        <div class="box">This is box-2</div>
        <div class="box">This is box-3</div>
        <div class="box">This is box-4</div>
        <div class="box">This is box-5</div>
        <div class="box">This is box-6</div>
        <div class="box">This is box-7</div>
        <div class="box">This is box-8</div>
        <div class="box">This is box-9</div>
        <div class="box">This is box-10</div>
    </div>
</body>
```

Now let us add background color and border to the box as follows-

```
.box{
        background-color: red;
        border: 2px solid black;
    }
```

The next step is to display the grids and decide the number of rows in it and then customize it as follows-

```
.grid{
        display: grid;
        grid-template-rows: 1fr 1fr 4fr;
        grid-auto-rows: 2fr;
        grid-template-columns: 1fr 4fr 2fr;
        grid-gap: 1rem;
    }
```

The grid-template-rows property is used to divide the first, second, and third rows in given fractions. The other rows will be set as two times the normal size. It is done with the help of grid-auto-rows. The grid-gap is used to decide the gap between the two grids. Lastly, the grid-template-columns is used to decide the number of columns used in the grid.

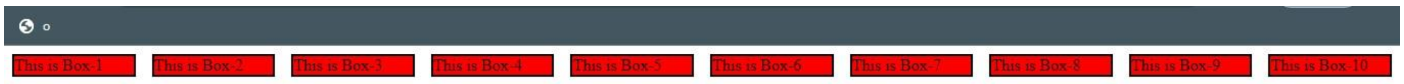
However, till now if we notice, these grid systems are not responsive which may leave a bad user experience. Therefore, we can make the use of repeat function which will fix the number of columns in the grid as follows-

```
<style>
.grid{
        display: grid;
        grid-template-rows: 1fr 1fr 4fr;
        grid-auto-rows: 2fr;
        grid-template-columns: repeat(4, 2fr);
        grid-gap: 1rem;
    }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Grid creating rows and gaps</title>
    <style>
        .grid{
            display: grid;
            /* there the starting 3 rows will covered the space of 1fr 1fr and 4fr and the
other left rows will be 1fr  */
            grid-template-rows: 1fr 1fr 4fr;
            /* grid-auto will give the left rows 3fr  */
            grid-auto-rows: 3fr;
            /* this will distribute the grid in two columns, 1 will be 1fr and other will
be 4fr  */
            grid-template-columns: 1fr 4fr;
            /* we can also distribute grids in columns like this  */
            grid-template-columns: repeat(10,2fr);
            /* grid-gap is used to give the gap between the grids  */
            grid-gap: 1rem;
        }
        .box{
            background-color: red;
            border: 2px solid black;
        }
    </style>
</head>
<body>
    <div class="grid">
        <div class="box">This is Box-1</div>
        <div class="box">This is Box-2</div>
        <div class="box">This is Box-3</div>
        <div class="box">This is Box-4</div>
        <div class="box">This is Box-5</div>
        <div class="box">This is Box-6</div>
        <div class="box">This is Box-7</div>
        <div class="box">This is Box-8</div>
        <div class="box">This is Box-9</div>
        <div class="box">This is Box-10</div>
    </div>
</body>
</html>
```

Output:



```
This is Box-1  This is Box-2  This is Box-3  This is Box-4  This is Box-5  This is Box-6  This is Box-7  This is Box-8  This is Box-9  This is Box-10
```

Spanning Multiple Rows and Columns in Grid

For our HTML code, we will make 8 divs of items and give them the class box. So the HTML code is as follows-

```
<body>
    <div class="container">
```

```
        <div class="box">Item-1</div>
        <div class="box">Item-2</div>
        <div class="box">Item-3</div>
        <div class="box">Item-4</div>
        <div class="box">Item-5</div>
        <div class="box">Item-6</div>
        <div class="box">Item-7</div>
        <div class="box">Item-8</div>
    </div>
</body>
```

Now we will add modify the box class with CSS as follows-

```
.box{
        border: 2px solid black;
        background-color: rgb(228, 188, 228);
        padding: 23px;
}
```

Then we will add some CSS in the class container to make the grid as follows-

```
.container{
        display: grid;
        grid-template-columns: repeat(5, 1fr);
        grid-template-rows: repeat(4, 1fr);
        grid-gap: 1rem;
}
```

To make it look better, we can add a grid-gap in the grid. We can add two types of grid gaps in our grid; one for rows and other for columns as follows-

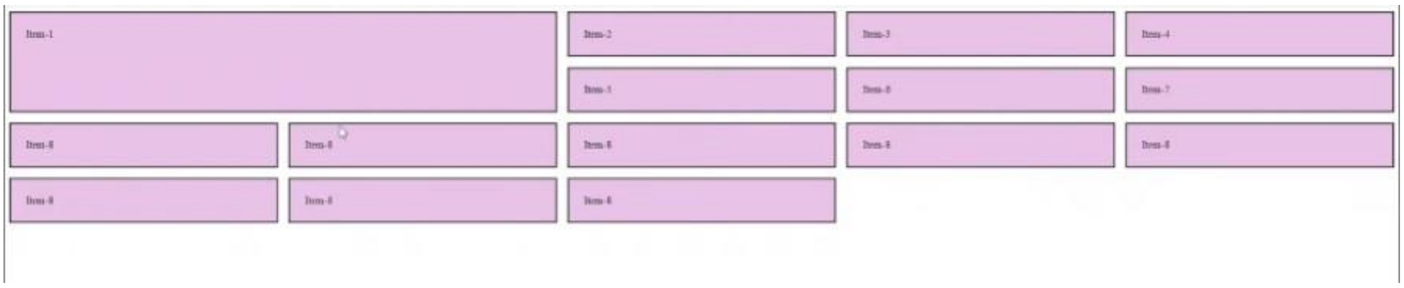
```
grid-column-gap: 7rem;
grid-row-gap: 1rem;
```

To understand better the concept of spanning, we can increase the number of items in the HTML. So after increasing the number of items we want that item of numbers 1,2,6 and 7 should we treated as one block.

For doing this we can use the property called grid row start and end. If we write the code as follows-

```
.box:first-child{
        grid-column-start: 1;
        grid-column-end: 3;
        grid-row-start: 1;
        grid-row-end: 3;
}
```

Result will be as follow:



With the help of this property, we can create extremely good layouts for our websites and it will be going to benefit us a lot in the future also for designing sidebars on the website.

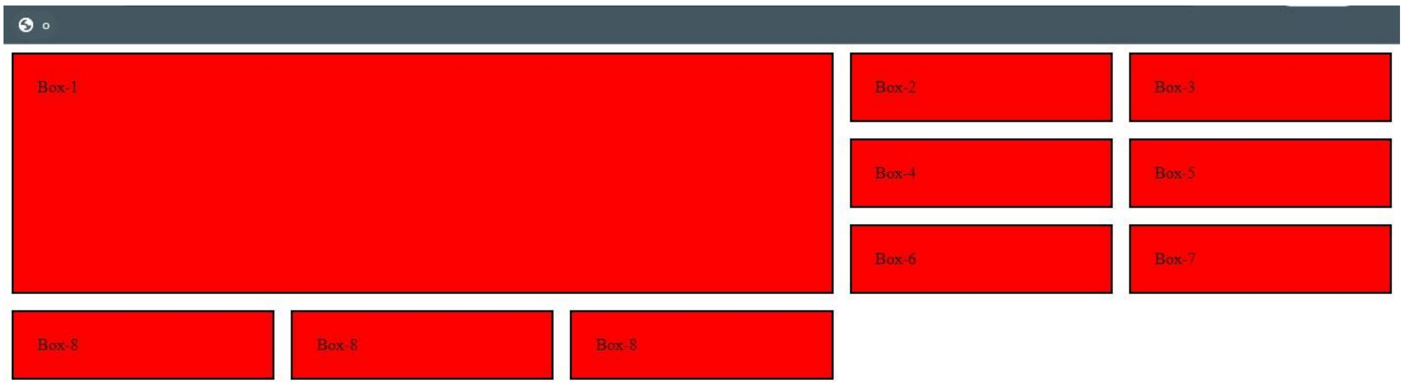
However, for the above code, there is a shortcut method also. It does not allow you to write the long code as above. The code is as follows-

```
box:first-child{
        /* grid-column-start: 1;
        grid-column-end: 3;
        grid-row-start: 1;
        grid-row-end: 3; */
        grid-column: 1 / span 3;
        grid-row: 1 / span 3;
    }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Spanning Multiple Rows and Columns</title>
    <style>
        .container{
            display: grid;
            grid-template-columns: repeat(5, 1fr);
            grid-template-rows: repeat(4, 1fr);
            /* to set the column gap of grid  */
            grid-column-gap: 2rem;
            /* to set the row gap of grid  */
            grid-row-gap: 1rem;
            grid-gap: 1rem;
        }
        .box{
            border: 2px solid black;
            background-color: red;
            padding: 23px;
        }
        .box:first-child{
            /* grid start and grid end (from 1 to 3 : this will span 2 columns)  */
            grid-column-start: 1;
            grid-column-end: 3;
            /* grid start and grid end (from 1 to 3 : this will span 2 rows)  */
            grid-row-start: 1;
            grid-row-end: 3;
            /* grid column (1/span 3) : this will span the 3 columns   */
            grid-column: 1/span 3;
            /* grid row (1/span 3) : this will span the 3 rows   */
            grid-row: 1 /span 3;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box">Box-1</div>
        <div class="box">Box-2</div>
        <div class="box">Box-3</div>
        <div class="box">Box-4</div>
        <div class="box">Box-5</div>
        <div class="box">Box-6</div>
        <div class="box">Box-7</div>
        <div class="box">Box-8</div>
        <div class="box">Box-8</div>
        <div class="box">Box-8</div>
    </div>
</body>
</html>
```

Output:



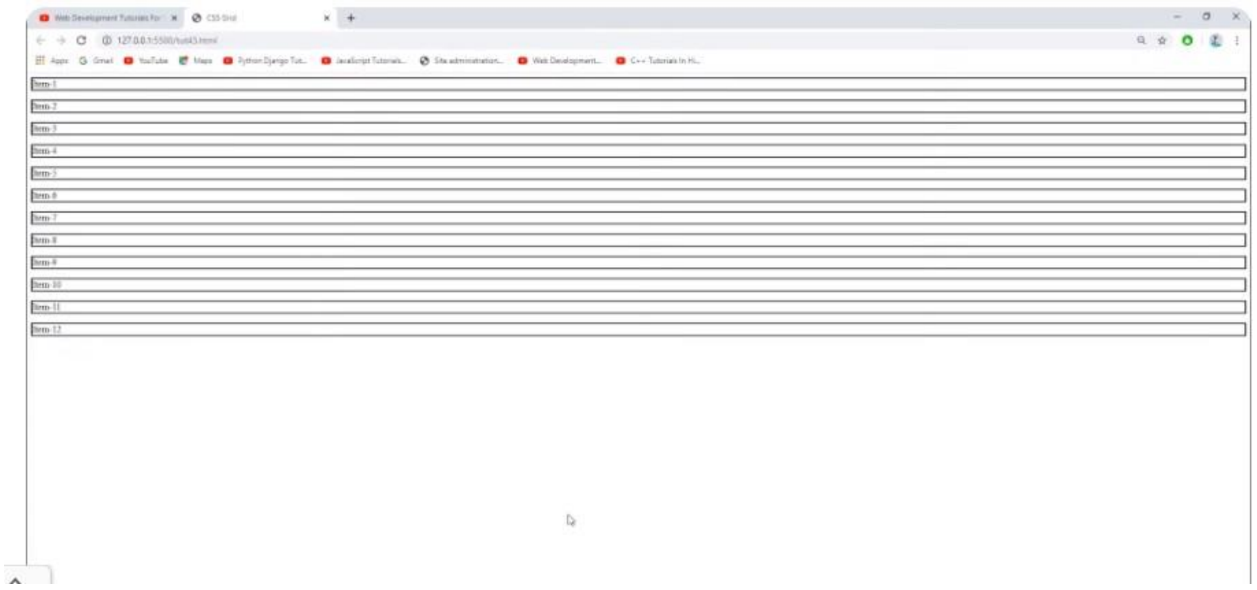
CSS Grid: Autofit & MinMax

```
<body>
        <div class="container">
        <div class="box">Item-1</div>
        <div class="box">Item-2</div>
        <div class="box">Item-3</div>
        <div class="box">Item-4</div>
        <div class="box">Item-5</div>
        <div class="box">Item-6</div>
        <div class="box">Item-7</div>
        <div class="box">Item-8</div>
        <div class="box">Item-9</div>
        <div class="box">Item-10</div>
        <div class="box">Item-11</div>
        <div class="box">Item-12</div>
    </div>
</body>
```

Now we can add some CSS in the container and boxes as follows-

```
.container{
            display: grid;
            grid-gap: 1rem;
}
.box{
            border: 2px solid black;
}
```

After writing the above code, the output will look like-

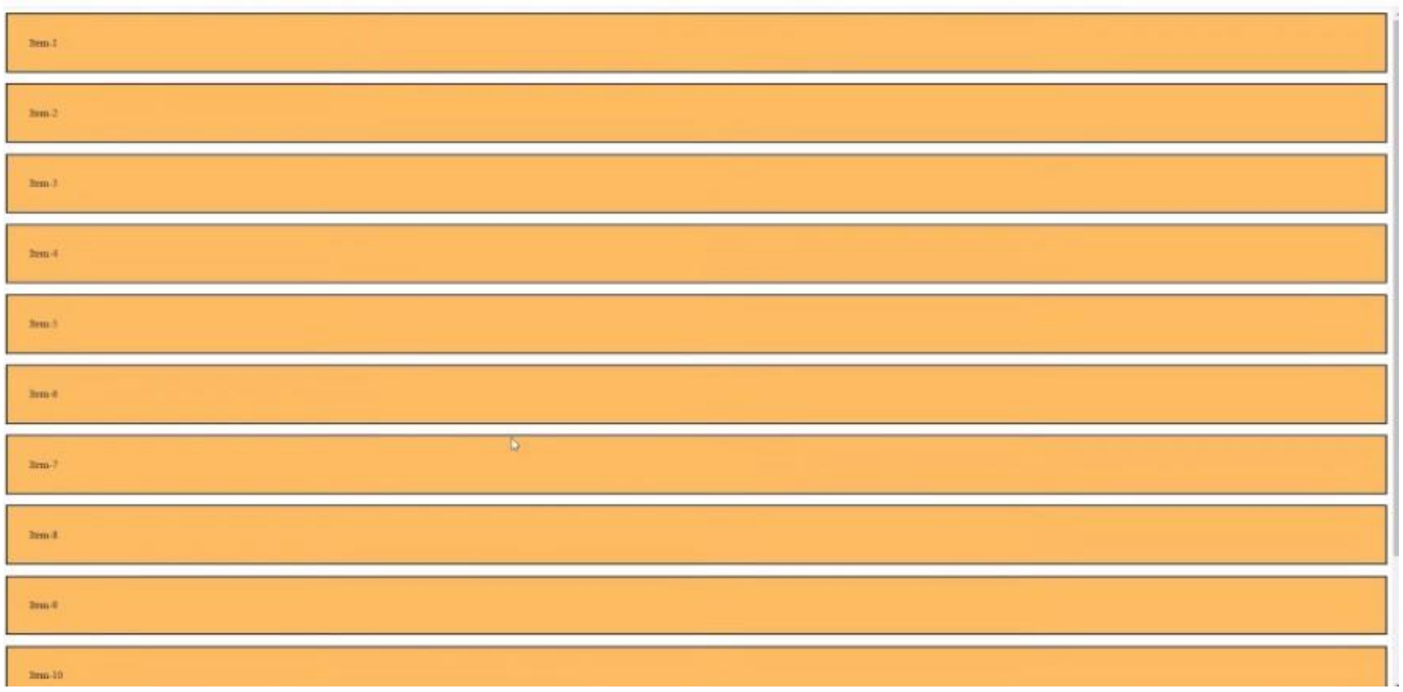


```
Item-1
Item-2
Item-3
Item-4
Item-5
Item-6
Item-7
Item-8
Item-9
Item-10
Item-11
Item-12
```

Now to give them the layouts of our choice, we can modify them by giving a background color and padding as follows-

```
.box{
            border: 2px solid black;
            background-color: rgb(245, 187, 101);
            padding: 34px;
        }
```

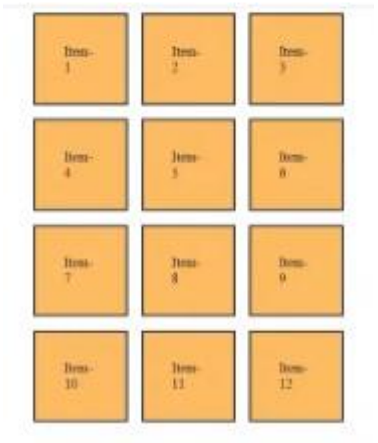
The output will look as follows-



If we modify the container as follows-

```
.container{
            display: grid;
            grid-gap: 1rem;
            grid-template-columns: 112px 112px 112px;
            justify-content: center;
        }
```

By adding the grid-template-columns, and setting the justify-content as the center, the output will look as follows-



Also, if we set the grid-template-columns as 1fr as follows-

```
grid-template-columns: 1fr 1fr 1fr;
```

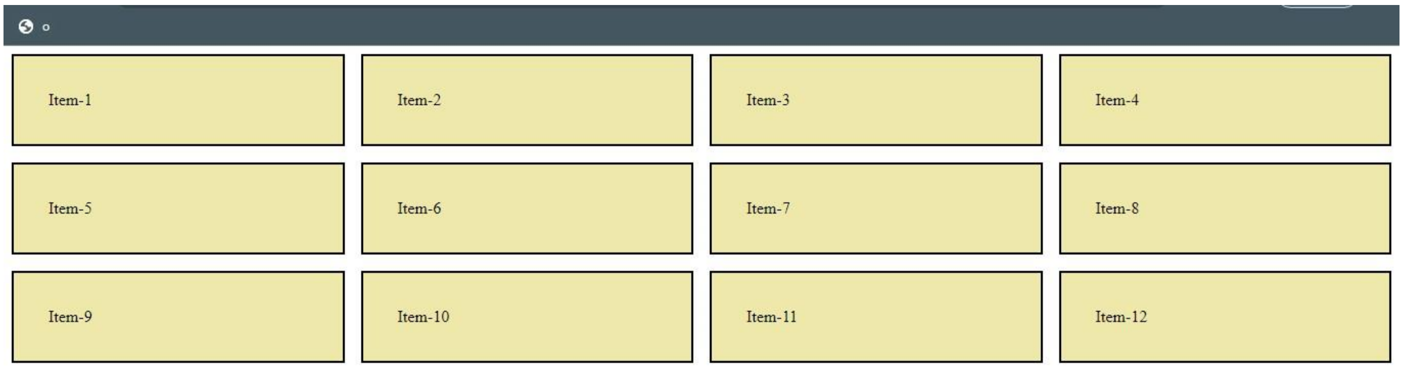
It will only set all the grids in 3 columns irrespective of the screen size. But if we want that all the grids should behave according to the screen size and adjust its size according to different devices, then we have to make them responsive. It means if the user is viewing our website on a mobile phone, he should see the number of rows less as compared while he is watching on a larger screen. So for that, we have to define our container as follows-

```
.container{
            display: grid;
            grid-gap: 1rem;
            /* grid-template-columns: 112px 112px 112px;  */
            /* grid-template-columns: 1fr 1fr 1fr;  */
            grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
            /* justify-content: center;  */
        }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Autofit and Minmax</title>
    <style>
        .container{
            display: grid;
            grid-gap: 1rem;
            /* grid-template-columns: 112px 112px 112px; */
            /* grid-template-columns: 1fr 1fr 1fr; */
            grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
            /* justify-content: center; */
        }
        .box{
            border: 2px solid black;
            background-color: palegoldenrod;
            padding: 34px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box">Item-1</div>
        <div class="box">Item-2</div>
        <div class="box">Item-3</div>
        <div class="box">Item-4</div>
        <div class="box">Item-5</div>
        <div class="box">Item-6</div>
        <div class="box">Item-7</div>
        <div class="box">Item-8</div>
        <div class="box">Item-9</div>
        <div class="box">Item-10</div>
        <div class="box">Item-11</div>
        <div class="box">Item-12</div>
    </div>
</body>
</html>
```

Output:



Creating Layouts Using Grid Template Area

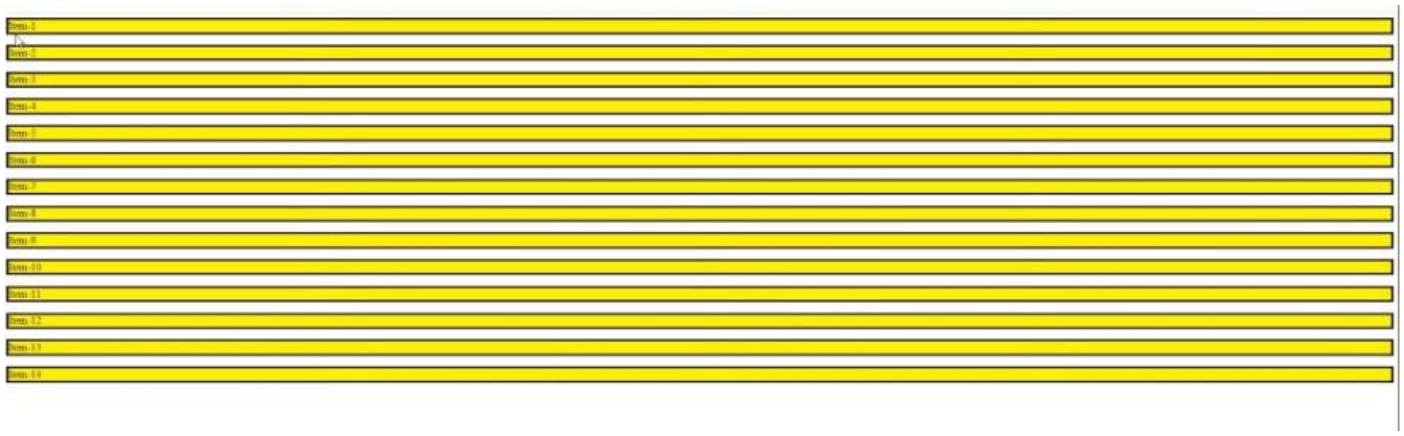
Now, Let’s start with the code

```
<body>
    <div class="container">
        <div class="item">Item-1</div>
        <div class="item">Item-2</div>
        <div class="item">Item-3</div>
        <div class="item">Item-4</div>
        <div class="item">Item-5</div>
        <div class="item">Item-6</div>
        <div class="item">Item-7</div>
        <div class="item">Item-8</div>
        <div class="item">Item-9</div>
        <div class="item">Item-10</div>
        <div class="item">Item-11</div>
        <div class="item">Item-12</div>
        <div class="item">Item-13</div>
        <div class="item">Item-14</div>
    </div>
</body>
```

Now we have to customize our container and items class by adding some CSS to it as follows-

```
container{
        display: grid;
        grid-gap: 1rem;
}
.item{
        background-color: yellow;
        border: 3px solid black;
        padding: 12px 23px;
}
```

The result will look as follows-



To understand better the concept of grid template areas, we can create new divs under the container class with id navbar, section, and aside as follows-

```
<div class= “container”>
          <div id= “navbar” class= “item”></div>
          <div id= “section” class= “item”></div>
          <div id= “section” class= “item”></div>
</div>
```

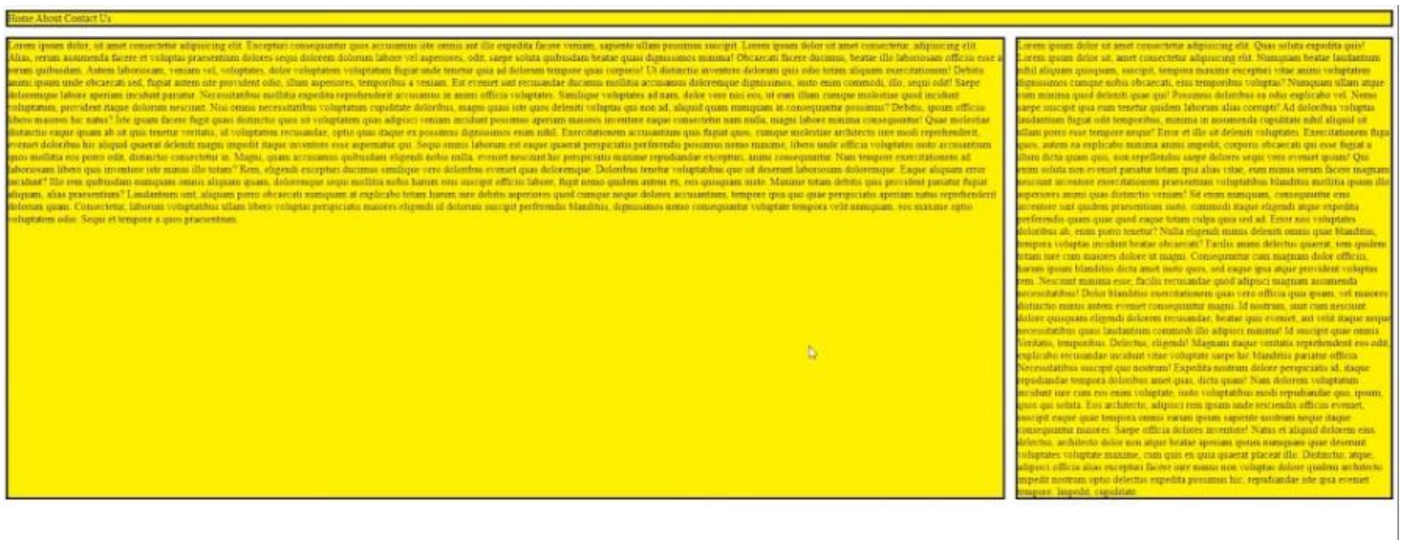
Now we can add CSS to all these three id’s as follows-

```
#navbar{
        grid-area: navbar;
    }
    #section{
        grid-area: section;
    }
    #aside{
        grid-area: aside;
    }
```

Till now only half of the work is completed. To understand completely about the grid template areas, we simply have to apply some CSS in the container. We simply have to add a matrix in the grid with some rows or columns. A row should contain the same number of columns as follows-

```
.container{
        display: grid;
        grid-gap: 1rem;
        grid-template-areas:
        'navbar navbar navbar navbar'
        'section section section aside'
}
```

Now the result of the above code will look as follows-



To create very simple websites, grids are highly recommendable and they can benefit us in lot of other things as well. For example, if we want to create a footer on the website, it can be easily done through grids as follows-

```
<footer class="item">Lorem, ipsum dolor sit amet consectetur adipisicing elit. Provident quo libero cumque.</footer>
```

To add CSS, we can write-

```
footer{
        grid-area: footer;
    }
```

We also have to update the container with 4 footers as follows-

```
.container{
        display: grid;
        grid-gap: 1rem;
        grid-template-areas:
        'navbar navbar navbar navbar'
        'section section section aside'
        'footer footer footer footer ';
    }
```

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Layouts using grid</title>
    <style>
```

```
        .container{
            display: grid;
            grid-gap: 1rem;
            /* setting areas for rows and columns  */
            grid-template-areas:
            /* There are 3 rows and 4 columns  */
            /* navbar will cover 4 columns of 1st row */
            'navbar navbar navbar navbar'
            /* section will cover starting 3 columns of 2nd row and aside will cover 4th
column of 2nd row  */
            'section section section aside'
            /* sanjay will cover 4 columns of 3rd row */
            'sanjay sanjay sanjay sanjay'
            ;
        }
        .box{
            background-color: orange;
            border: 3px solid black;
            padding: 12px 23px;
        }
        #navbar{
            /* we are setting the name of grid area for call it in grid-template-area  */
            grid-area: navbar;
        }
        #section{
            grid-area: section;
        }
        #aside{
            grid-area: aside;
        }
        #footer{
            text-align: center;
            grid-area: sanjay;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box" id="navbar">HOME ABOUT CONTACT US</div>
        <div class="box" id="section">Lorem ipsum dolor sit amet, consectetur adipisicing
elit. Itaque obcaecati iure, porro eaque consequuntur harum nostrum deleniti. Blanditiis,
eaque nesciunt?</div>
        <div class="box" id="aside">Lorem ipsum dolor sit amet consectetur adipisicing
elit. Animi, facilis atque! Tempora odio fuga rem amet laudantium dolor omnis
facere!</div>
        <div class="box" id="footer">FOOTER</div>
        <!-- <div class="box">Item-1</div>
        <div class="box">Item-2</div>
        <div class="box">Item-3</div>
        <div class="box">Item-4</div>
        <div class="box">Item-5</div>
        <div class="box">Item-6</div>
        <div class="box">Item-7</div>
        <div class="box">Item-8</div>
        <div class="box">Item-9</div>
        <div class="box">Item-10</div>
        <div class="box">Item-11</div>
        <div class="box">Item-12</div>
        <div class="box">Item-13</div>
        <div class="box">Item-14</div> -->
    </div>
</body>
</html>
```

Media Queries with CSS Grid

We will use the concept of media queries which we have already discussed in the previous project. After applying the media queries, we will check whether those are applicable or not by adding different colors as follows-

```
@media only screen and (max-width:300px) {
        body {
            background-color: red;
        }
@media only screen and (min-width: 300px) and (max-width:500px) {
        body {
            background-color: blue;
        }
@media (min-width: 500px) and (max-width:800px) {
        body {
            background-color: yellow;
        }
@media (min-width: 800px) {
        body {
            background-color: green;
        }
    }
```

The result of the above code will be, the color of the screen will automatically change depending upon the screen size. It means the media queries are working.

Now we will add the HTML code to get started as follows-

```
<body>
    <div class="container ">
        <nav class="bdr">
            <span>Home</span>
            <span>About</span>
            <span>Services</span>
            <span>Contact</span>
        </nav>
        <section class="bdr">
            <h2>Learn CSS in hindi</h2>
        </section>
        <aside class="bdr">
            <h1>More about us</h1>
        </aside>
    </div>
    <footer class="bdr">Copyright CodeWithHarry 2020</footer>
</body>
```

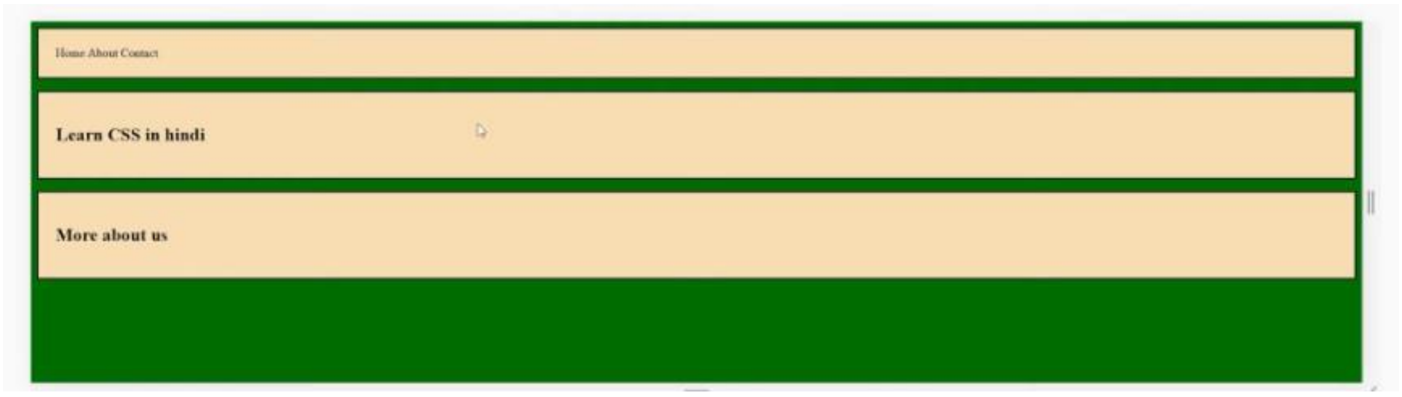
Now we will give the grid property to the container and a grid-gap to it as follows-

```
.container {
        display: grid;
        grid-gap: 1rem;
}
```

Also, we will give border, padding, and background color to it as shown below-

```
.bdr {
        border: 2px solid black;
        padding: 10px 23px;
        background-color: wheat;
    }
```

The result will look as follows-


Home About Contact

Learn CSS in hindi

More about us

Now we will add the grid-template-area property to the container and define the property grid area to nav, section, aside, and footer as follows-

```
grid-template-areas:
            'navbar navbar navbar navbar'
            'section section section aside'
            'footer footer footer footer ';
```

```
nav {
        grid-area: navbar;
    }
    section {
        grid-area: section;
    }
    aside {
        grid-area: aside;
    }
    footer {
        grid-area: footer;
        text-align: center;
    }
```

Now to make it responsive, we have to modify the container when the website is between 500px to 800px as follows-

```
.container {
            grid-template-areas:
                'navbar navbar navbar navbar'
                'section section section section'
                'aside aside aside aside'
                'footer footer footer footer ';
        }
```

Now when the screen size goes between 300px to 500px, we have to again modify the container to make it responsive as follows-

```
.container {
            grid-template-areas:
                'navbar navbar navbar navbar'
                'section section section section'
                'aside aside aside aside'
                'footer footer footer footer ';
        }
```

If we want to remove the aside tag when the screen size is between 300-500 pixels, we can set the property display as none as follows-

```
aside{
            display: none;
        }
```

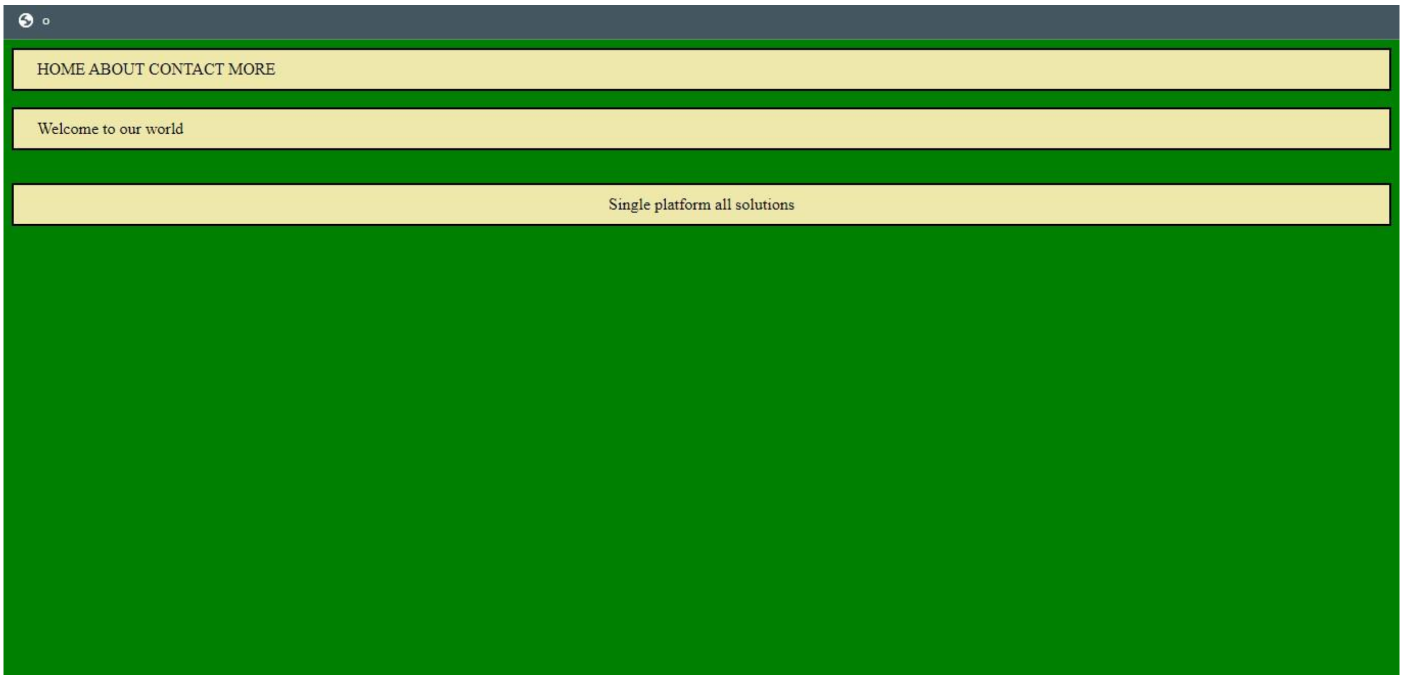
With media queries, you are capable of building any website of your choice. Therefore, you should practice more these media queries and build your own professional website to get command over.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Media Queries</title>
    <style>
        .container{
            display: grid;
            grid-gap: 1rem;
            grid-template-areas:
            'navbar navbar navbar navbar'
            'section section section aside'
            'sanjay sanjay sanjay sanjay'
        }
        @media only screen and (max-width:300px) {
            body{
                background-color: red;
            }
            .container{
            grid-template-areas:
            'navbar navbar navbar navbar'
            'section section section section'
            'aside aside aside aside'
            'sanjay sanjay sanjay sanjay';
                }
            span{
                display: block;
            }
            aside{
                    display: none;
                }
        }
        @media only screen and (min-width:300px) and (max-width:500px){

            body{
                background-color: yellow;

            }
            .container{
            grid-template-areas:
            'navbar navbar navbar navbar'
            'section section section section'
            'aside aside aside aside'
            'sanjay sanjay sanjay sanjay';
                }
        }
        @media (min-width:500px) and (max-width:800px){
            body{
                background-color: pink;
            }
        }
        @media (min-width:800px){
            body{
                background-color:green ;
            }
            .container{
            grid-template-areas:
            'navbar navbar navbar navbar'
            'section section section section'
            'aside aside aside aside'
            'sanjay sanjay sanjay sanjay';
```

```
                }
            aside{
                    display: none;
                }
        }
        #navbar{
            /* we are setting the name of grid area for call it in grid-template-area  */
            grid-area: navbar;
        }
        #section{
            grid-area: section;
        }
        #aside{
            grid-area: aside;
        }
        #footer{
            text-align: center;
            grid-area: sanjay;
        }
        .bdr{
            padding: 10px 23px;
            border: 2px solid black;
            background-color: palegoldenrod;
        }
    </style>
</head>
<body>
   <div class="container">
    <nav class="bdr" id="navbar">
        <span>HOME</span>
        <span>ABOUT</span>
        <span>CONTACT</span>
        <span>MORE</span>
    </nav>
    <section class="bdr" id="section">
        Welcome to our world
    </section>
    <aside class="bdr" id="aside">
        I'm mr adroit
    </aside>
   </div>
   <footer class="bdr"  id="footer">Single platform all solutions</footer>
</body>
</html>
```

Output:



HOME ABOUT CONTACT MORE

Welcome to our world

Single platform all solutions