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

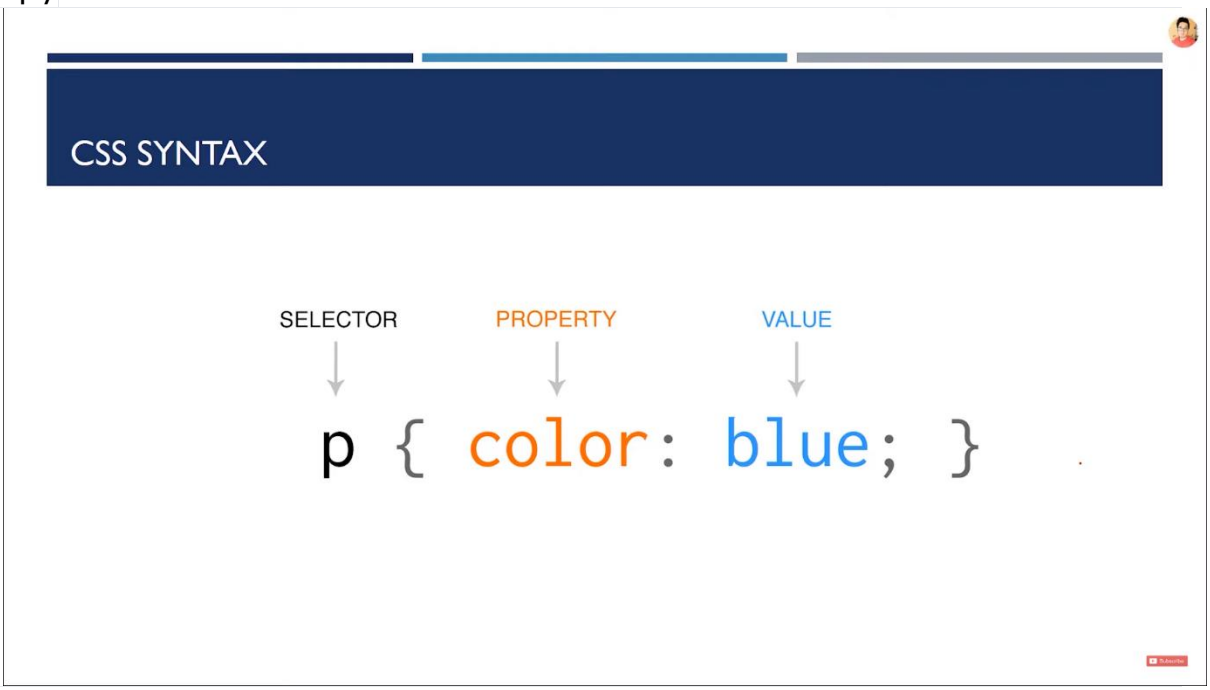
- CSS is a style sheet language that is used to handle the presentation of the web page containing HTML.
- It makes our websites beautiful and modern looking.

CSS Syntax

The syntax of CSS is-

```
P { color: blue; }
```

Copy

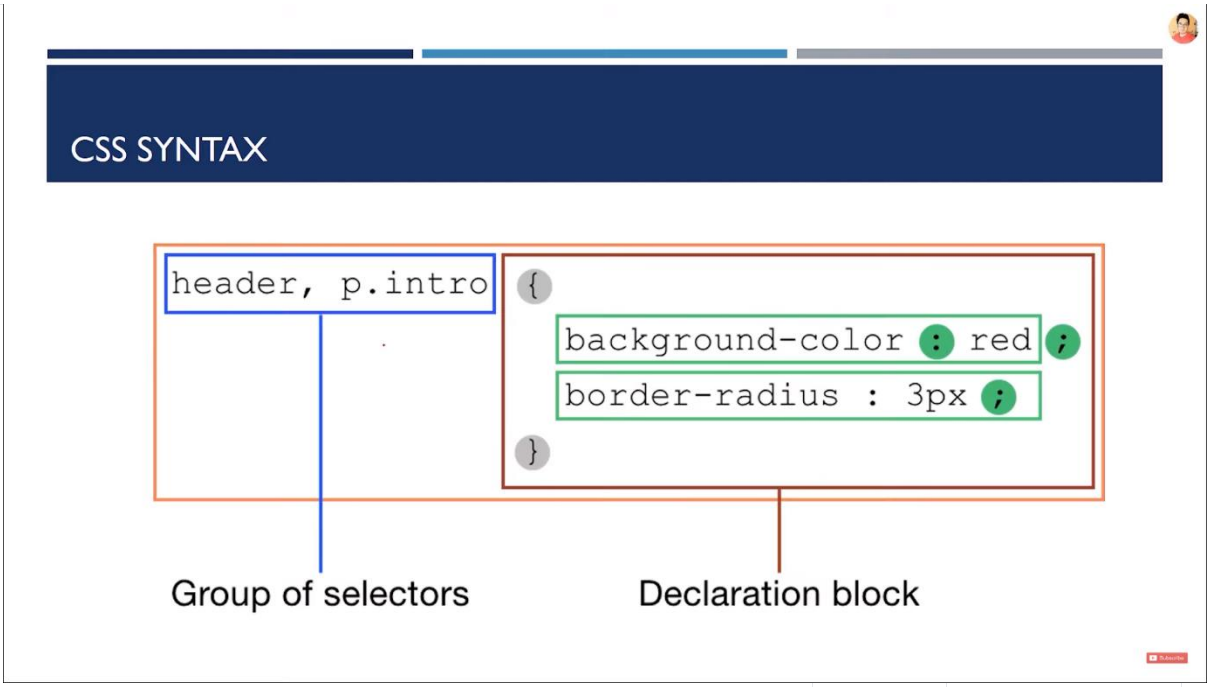


- *P* stands for the selector and it decides which part of the HTML the CSS will be applied. It states *where* the CSS property is to be applied.
- Property is used to describe which property you want to change or add. Whether you have to change colour, border, background, width, all these come under 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,
                                  }
```

Copy



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-

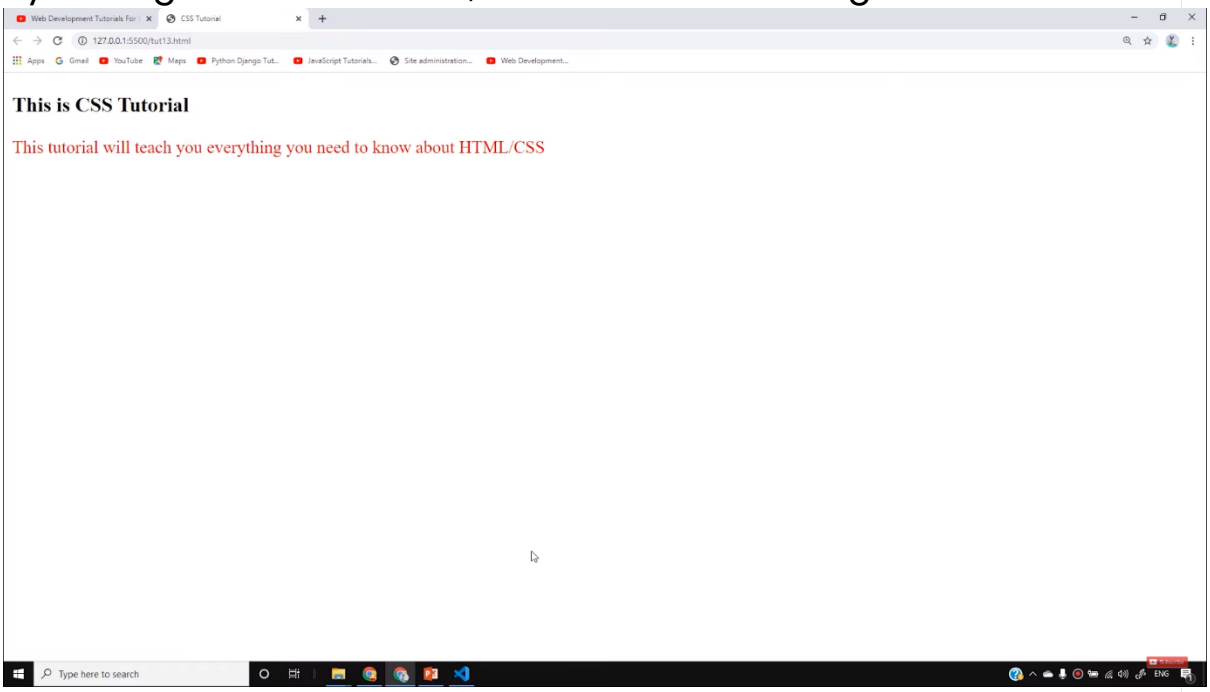
    -
    - First, write the CSS in.CSS file.
    - Include that CSS file to Markup.

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

Copy

By writing the above code, we will see the changes as-



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.

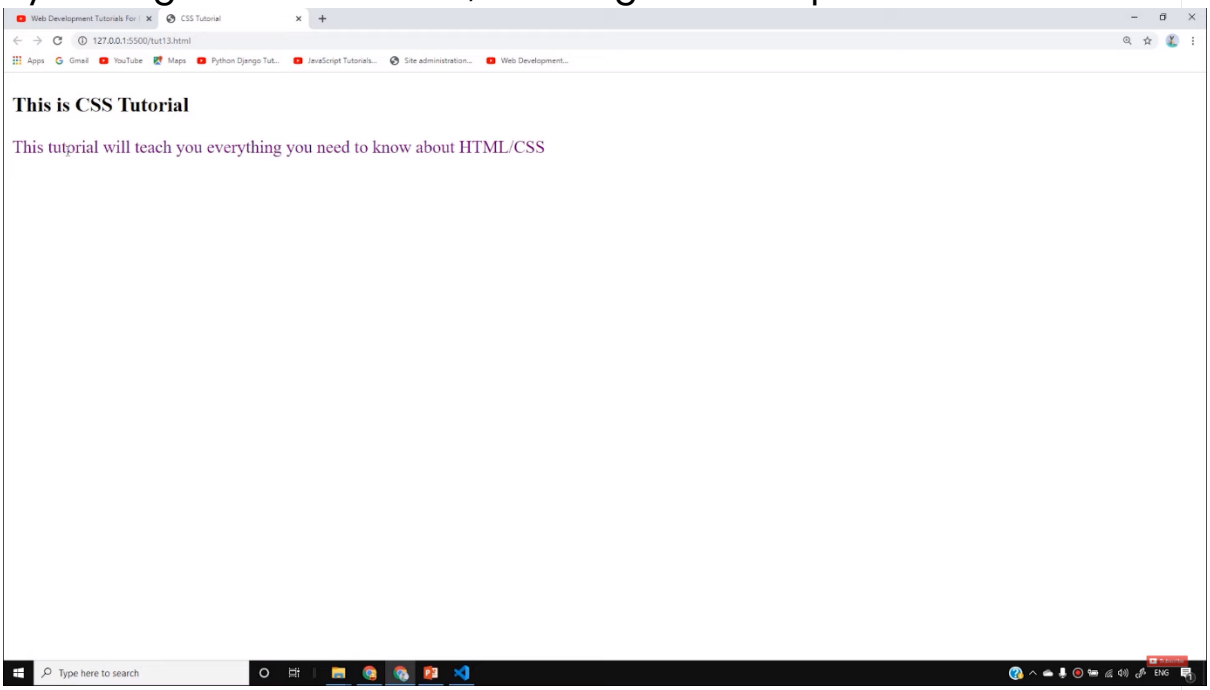
- **Internal CSS** is used to define a style tag for a single HTML page. It is defined in the <head> section within a <style> element. Let us understand the External CSS with the help of an example.

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

```
                }
    <style>
<head>
```

Copy

By writing the above code, we will get the output as-



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= “tut13.css”>
<head>
```

Copy

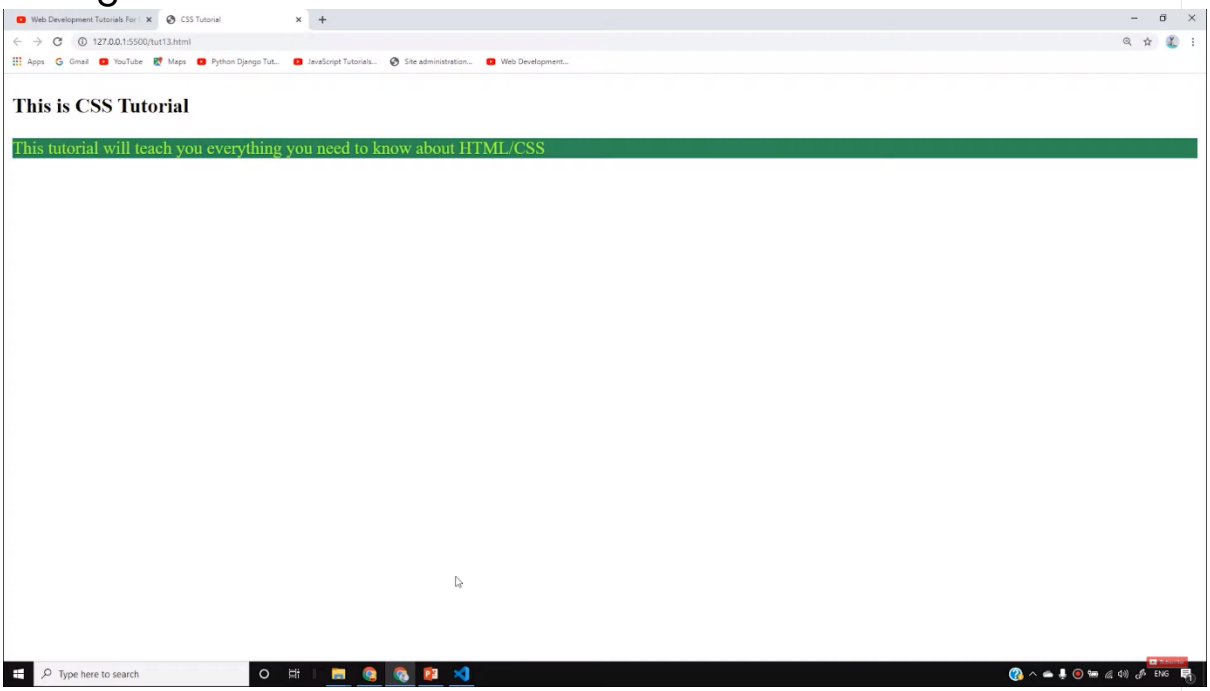
Then in *tut13.css* file, we can write our CSS-

```
p{
```

```
        Color: greenyellow;
}
```

Copy

By writing this code, and linking it to the HTML file, you notice the change-



The point here to remember is that whether it is the Internal CSS or the External CSS, whichever is written after, gets the priority. But if you want that first one should get priority, so you can add ***important*** after it. The result will be, it will get the most priority of all.

So I hope you must have understood all the three types of CSS used to style our website. From the next tutorials, we will see some more interesting properties about CSS. Till then stay with the tutorials and keep practicing whatever is taught till now.

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*

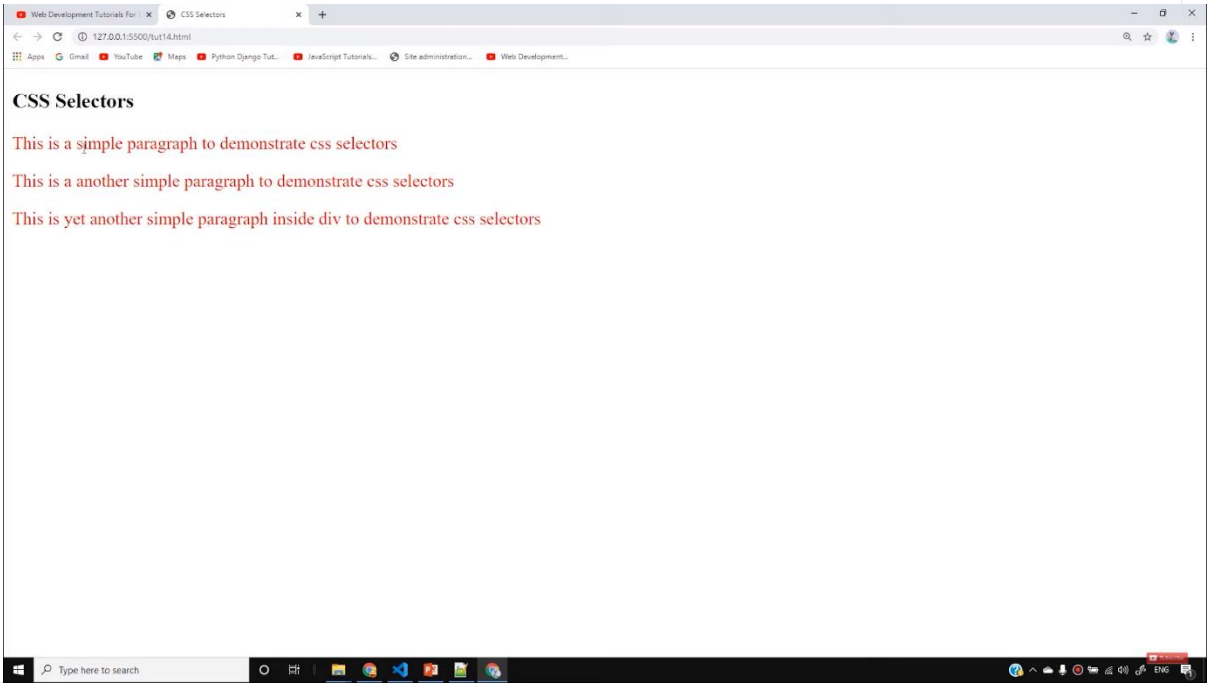
As discussed in one of the previous videos, the basic syntax of writing the CSS is-
p {color: blue;}
In the example above, 'p' is the selector. It will convert all the paragraph into blue.
So let us now start by making a new file as *tut14.html and as usual, add an instant boilerplate* Visual Studio 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>
```

Copy

The above code will convert all the three paragraphs into red colour as shown below-



- **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>
```

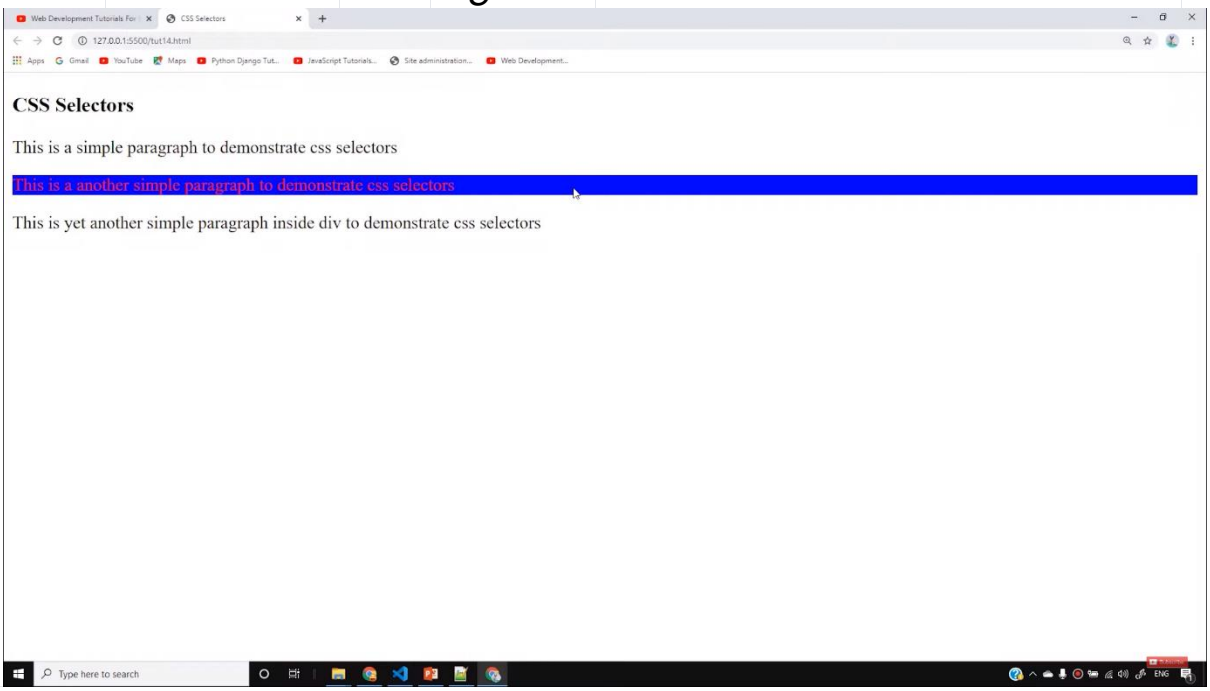
Copy

```
<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>
```

Copy

This will convert only the second paragraph with class *"redElement"* and *"bgBlue"* as shown below-



- **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;
                    }
```

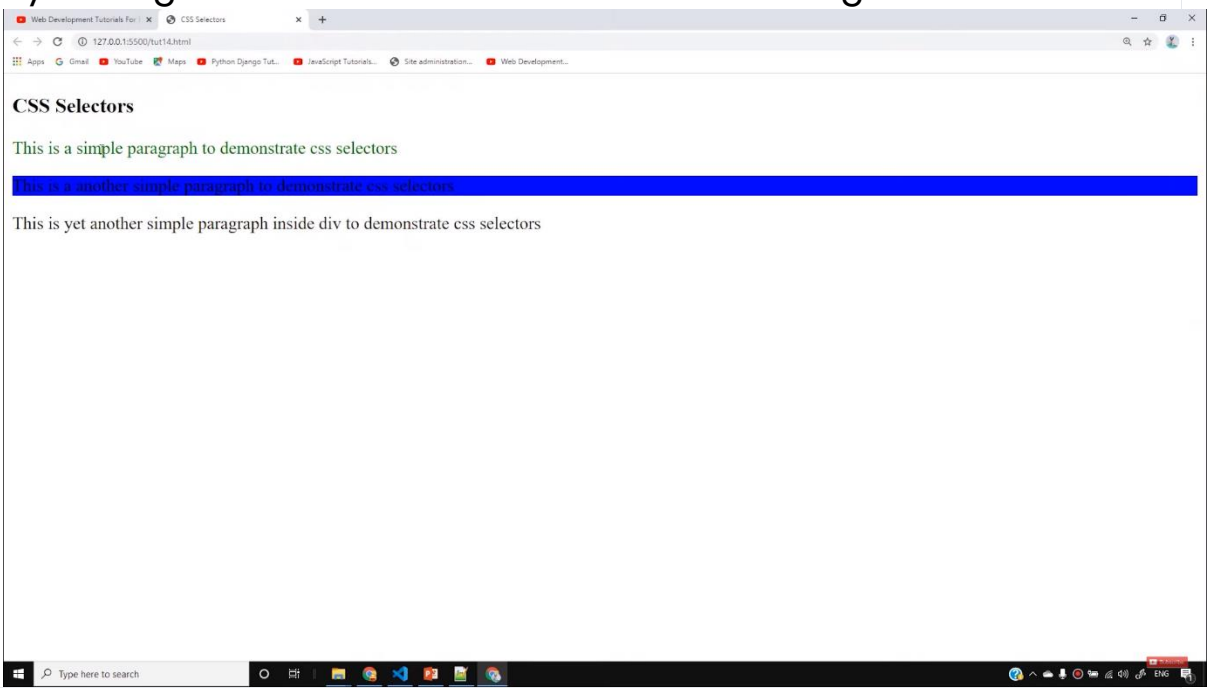
```
</style>
```

Copy

```
<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>
```

Copy

By writing the above code we will see the changes as follows-



- **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>
```

Copy

```
<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>
```

Copy

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 as described/written in the video

```
<!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 Selectors</title>
    <style>
        /* Element selector */
        p{
            border: 2px solid red;
        }

        /* Id selector */
        #firstPara{
            color: green;
        }

        /* Class selector */
        .bgBlue{
            color: yellow;
            background-color: blue;
        }

        /* Grouping selector */
        footer, span{
            background-color: pink;
        }
    </style>
</head>
<body>
    <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>
        <span>this is span</span>
    </div>
    <footer>This is footer</footer>
</body>
</html
```

Give the title as *Developer Tools* in the <title> tag. Let us start by writing the simple code-

```
<style>

        p{

            color: purple;

            font-style: italic;

            background-color: rosybrown;

        }

        .bgPrimary{

            background-color: #82c2ff;

        }

    </style>

</head>
```

```
<body>

    <h4 class="bgPrimary">Developer tools tutorial</h4>

    <p>This is a tutorial for Chrome developer tools</p>

</body>
```

Copy

After running your code, if you will right-click anywhere in the browser then you will see an **inspect element option.** By clicking on it you will be able to see the original code. From here you can make some changes in the webpage and can observe it. However, it does not change the original code on your server. Developer Tools are used to make any changes into your code and see the instant effect on your web page. This change is not a permanent.

But if you like any change made in the developer tools, you can do the same in original file in VS code and refresh the page. That particular change will now reflect back permanently. **Inspect Element** allows us to make and view the changes in any of the websites present all over the world

I have explained that how these changes are only reflecting on your local server. When you will reload that page again, all the things will set back to their default set up. In this way, you can use the developer tools of chrome.

User Agent Stylesheet

By default, chrome sets some property for some elements and store some default values in it. So the browser by default styles some element according to it and that styling is particularly known as user's agent style sheet. Basically it is showing the default value of browser that previously what changes were there and what changes you have made now on your

page. There is also a console section where you can write JavaScript code. Sources section contains all the source codes that you have written. The most important thing in this tutorial was **elements** that we have already discussed. We can directly change our styles using CSS directly.

Some tips while defining a color class:-

Do not directly use the name of the color as the class name instead use

- colorprimary
- colorsuccess
- colorwarning

As writing these looks more professional and also helps in future to maintain our codes.

So, I believe you must have learned something about Developer Tools and how to use them. The more practicing and playing around Inspect Elements will help you in learning faster.

Code as described/written in the video

```
<!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>Developer tools</title>
    <style>
        p{
            color: purple;
            font-style: italic;
            background-color: rosybrown;
```

```
        }
        .bgPrimary{
            background-color: #82c2ff;
        }
    </style>
</head>
<body>
    <h4 class="bgPrimary">Developer tools tutorial</h4>
    <p>This is a tutorial for Chrome developer tools</p>
</body>
</html>
```

CSS Fonts under the <title> tags in the <head> section.

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

Copy

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;}
```

Copy

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;
}
```

Copy

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;
    }
```

Copy

- 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;
}
```

Copy

There are various other font properties. Most of the important ones are covered in this tutorial. You can now test different other font properties as well. As a beginner, I would recommend not to learn all the CSS properties until you learn to make a simple website. You can take the help of references available anytime but for it, you should know the basics.

Code as described/written in the video

```
<!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 Fonts</title>
```

```
    <link href="https://fonts.googleapis.com/css?family=Ubuntu&display=swap" rel="stylesheet">
    <style>
        p{
            font-family:'Ubuntu', 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
            font-size: 33px; /* 1/96th of an inch */
            line-height: 1.3em;
        }

        span{
            font-weight: bold;
            font-style: italic;
        }
    </style>
</head>
<body>
    <h4>CSS Fonts</h4>
    <p>Lets play with <span>fonts</span>. It is very exciting</p>
</body>
</html>
```

Colors in CSS in <title> tag under the head section. The basic code for our example is-

```
<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>
```

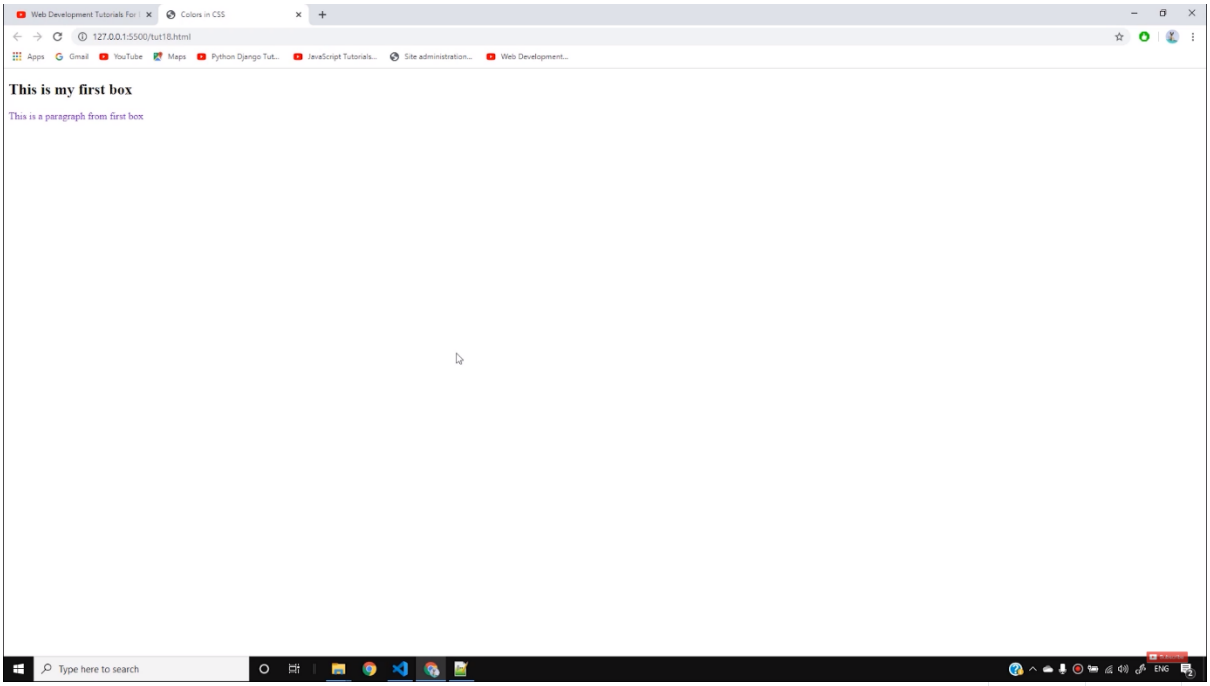
Copy

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

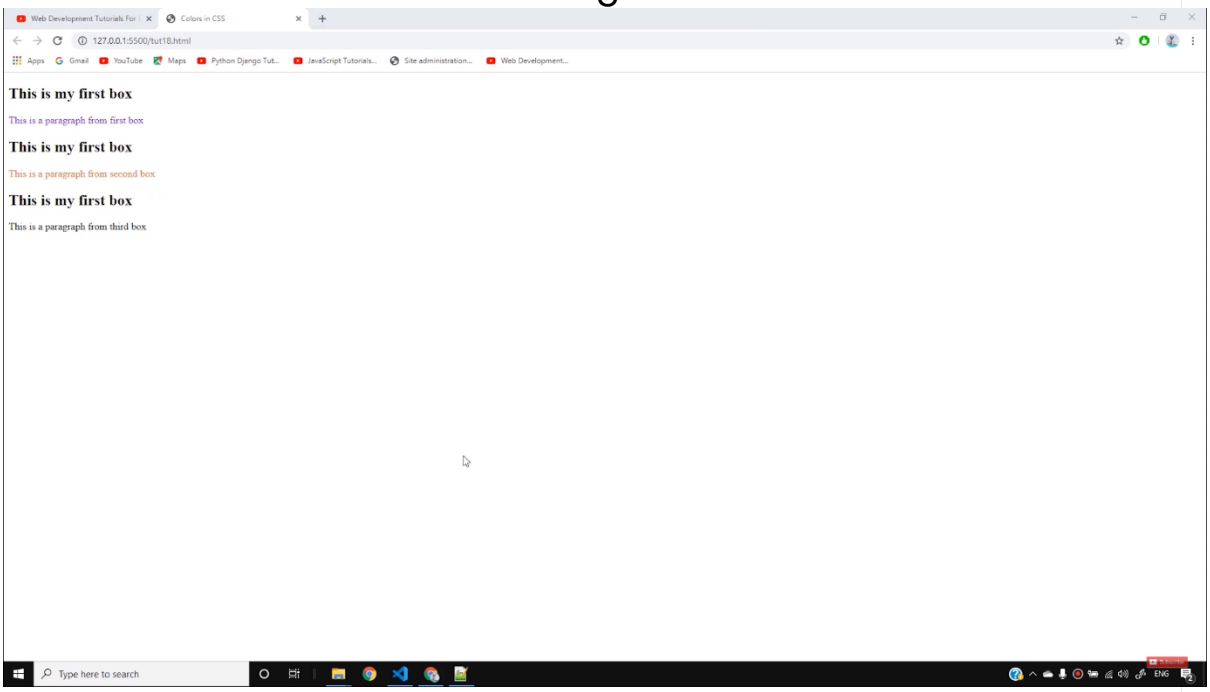
```
#firstPara{
            color:blueviolet; /* Color by name */
        }
```

Copy

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.

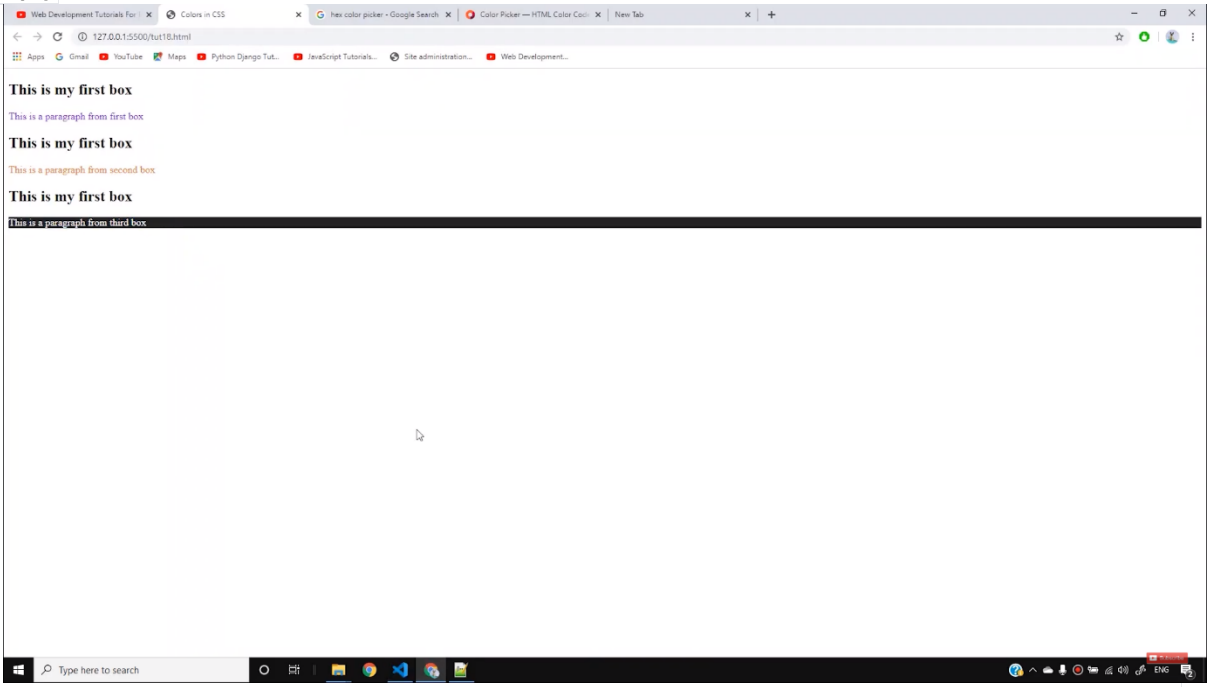


Here, we treat RGB as a function and pass the values in that function for red, green and blue. As we give different values in the function, it creates various combinations of different colours combined with red, green, and blue.

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

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

Copy



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:

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

Copy

Wherever the colour property is used in CSS, you can apply any of the above three methods. Certain more properties use colour, like border property and background colour, which we will learn step by step as we move ahead in building professional websites.

Code as described/written in the video

```
<!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>Colors in CSS</title>
    <style>
        #firstPara{
            color:blueviolet; /* Color by name */
        }

        #secondPara{
            color: rgb(223, 130, 54); /* Color by rgb value */
        }

        #thirdPara{
```

```
            color: white;
            background-color: #ff4532; /* Color by hex value */
        }
    </style>
</head>
<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>
</html>
```

Give the title as **Height, width, borders and backgrounds** in the <title> tag under the head section.
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>
```

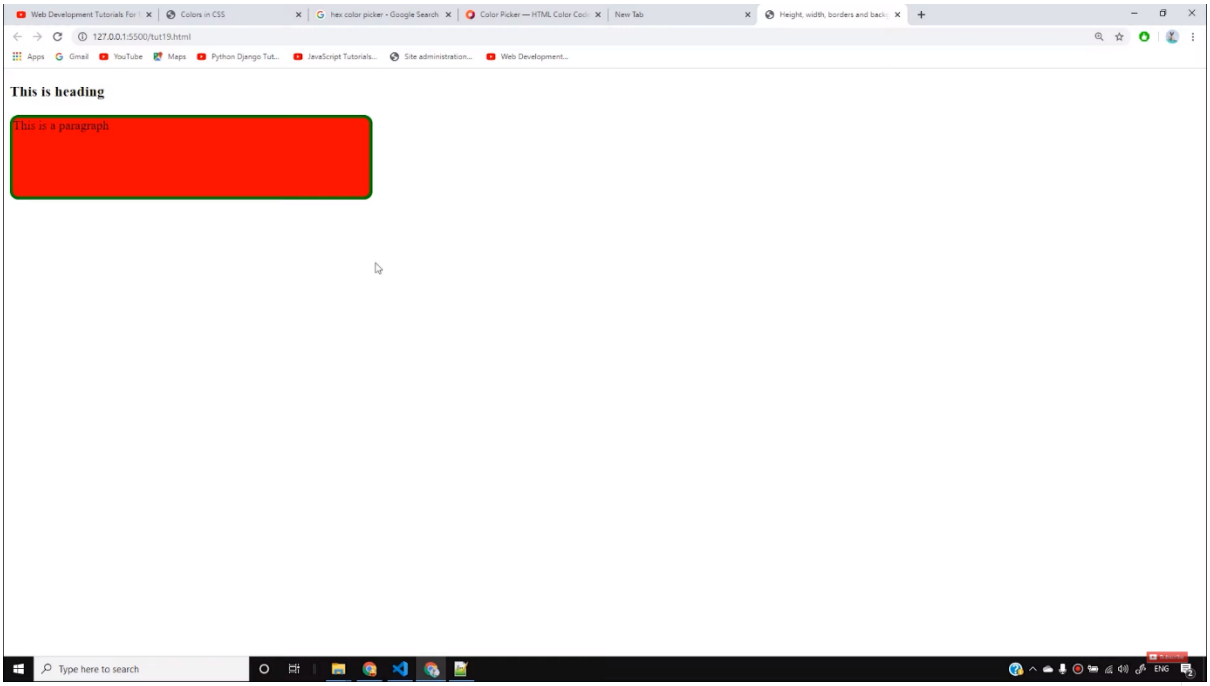
Copy

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;
        }
```

Copy

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 -



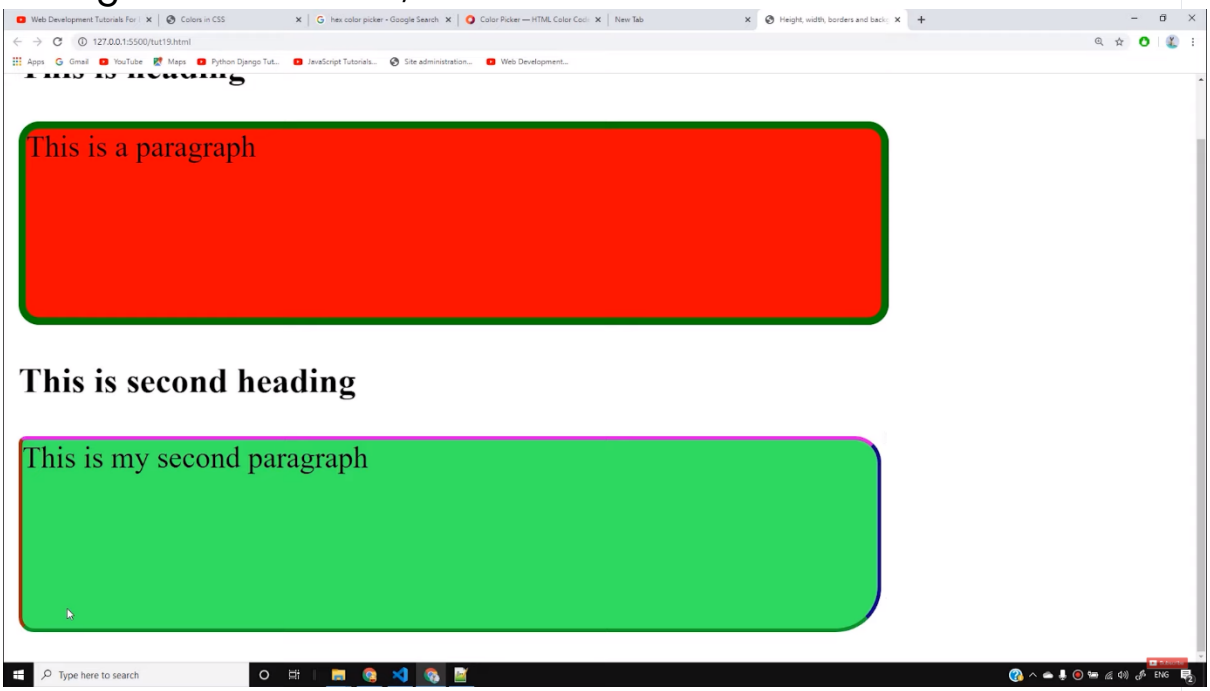
Now if the condition arises that you want to give a border only at one end, then what will you do. Let us understand with an example -

```
#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;
        }
```

Copy

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-

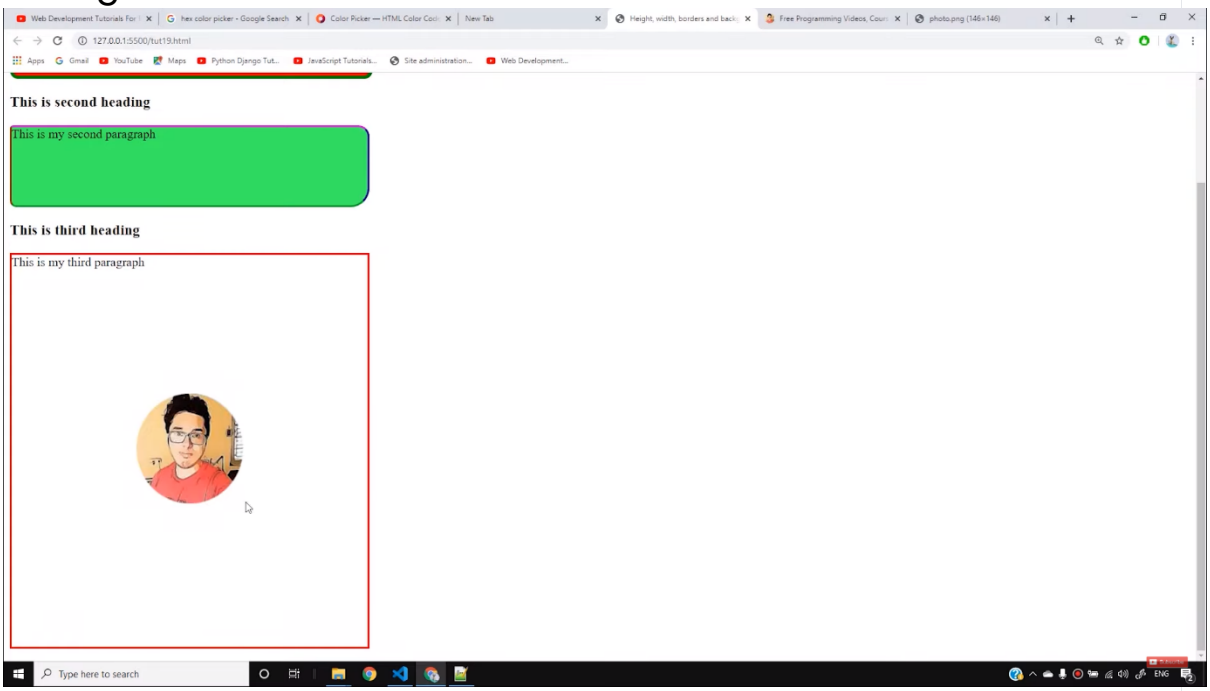


Now what if, if you want to add a background image behind the text that you have written. Let us understand this with the code -

```
#thirdPara{
            height: 500px;
            width:455px;
            background-image:
url('https://codewithharry.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; */
        }
```

Copy

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. You can see the changes made below-



So, I believe you must understood the concepts of different other properties related to borders and colors. You can try out different other combinations of these properties to get better knowledge.

Code as described/written in the video

```
<!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>Height, width, borders and backgrounds </title>
```

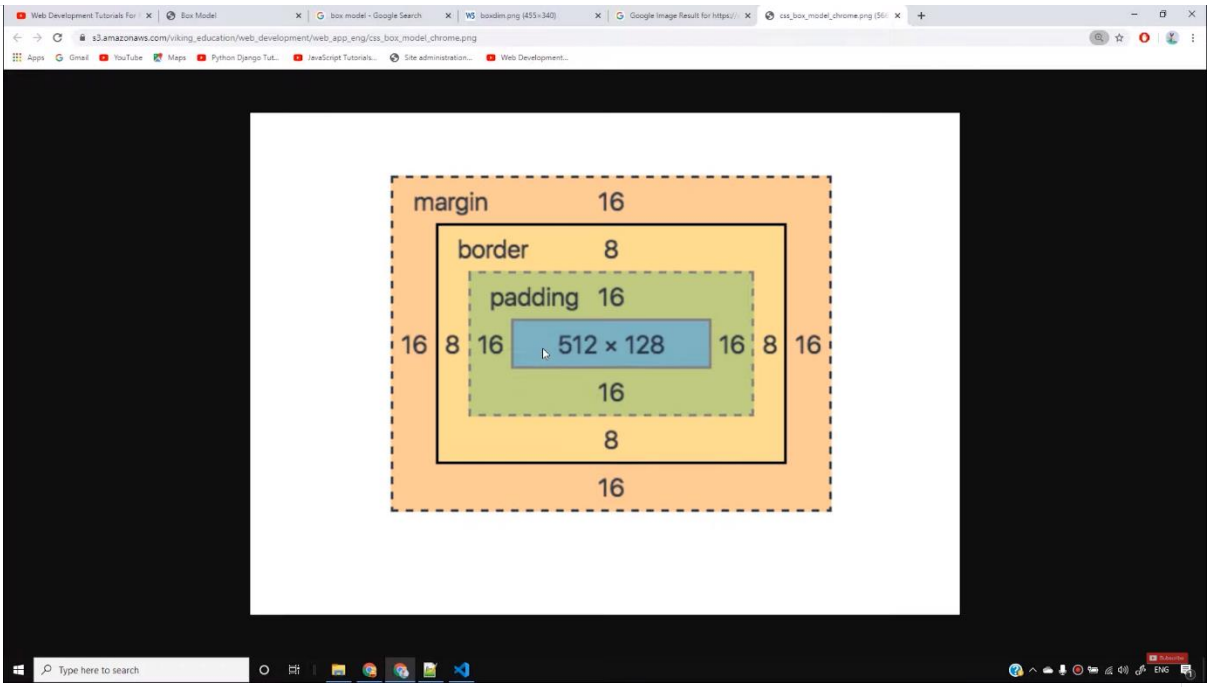
```
    <style>
        #firstPara{
            background-color: red;
            height: 100px;
            width:455px;
            border: 4px solid green;
            /* border-width: 4px;
            border-color: green;
            border-style: solid;  */
            border-radius: 11px;
   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

        #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;
        }

        #thirdPara{
            height: 500px;
            width:455px;
            background-image:
url('https://codewithharry.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; */

        }
    </style>
</head>
<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>
</html>
```

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
        }
```

Copy

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;
```

Copy

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;
```

Copy

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;
```

Copy

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;
```

Copy

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;
        }
```

Copy

So I believe the concept of CSS Box Model and Box Sizing is understood to you. To understand more deeply, you can try to change the values and see all the necessary changes.

Code as described/written in the video

```
<!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>Box Model</title>
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
        }
        body{
            background-color: #e6cbf8;
        }

        .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;



        }
    </style>
</head>
<body>
    <div class="container">
        <h3>This is my heading</h3>
        <p id="first">Lorem ipsum dolor sit amet consectetur, adipisicing elit. Incidunt harum quis, quibusdam, minima molestiae tempore vel magni, repellendus doloribus debitis rerum tenetur eveniet.</p>
    </div>

    <div class="container">
        <h3>This is my heading</h3>
        <p id="second">Lorem ipsum dolor sit amet consectetur, adipisicing elit. Incidunt harum quis, quibusdam, minima molestiae tempore vel magni, repellendus doloribus debitis rerum tenetur eveniet.</p>
    </div>

    <div class="container">
        <h3>This is my heading</h3>
```

```
        <p id="third">Lorem ipsum dolor sit amet consectetur,
adipisicing elit. Incidunt harum quis, quibusdam, minima molestiae
tempore vel magni, repellendus doloribus debitis rerum tenetur
eveniet.</p>
    </div>
</body>
</html>
```

Alignment in the <title> tag.
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-

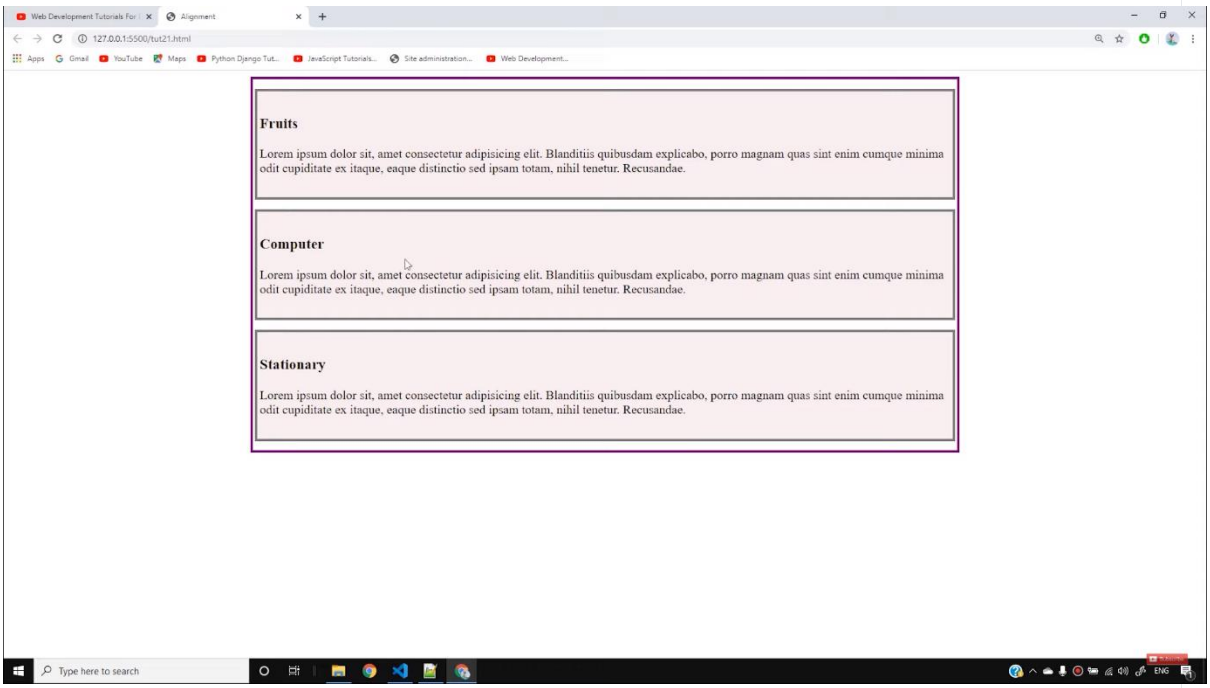
- **Left-** The elements floats to the left of its container.
- **Right-** The elements floats to the right of its container.
- **None-** The element does not float (it will be displayed just where it occurs in the text). This is default.
- **Inherit-** The element inherits the float value of its parent.

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);
        }
```

Copy

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-

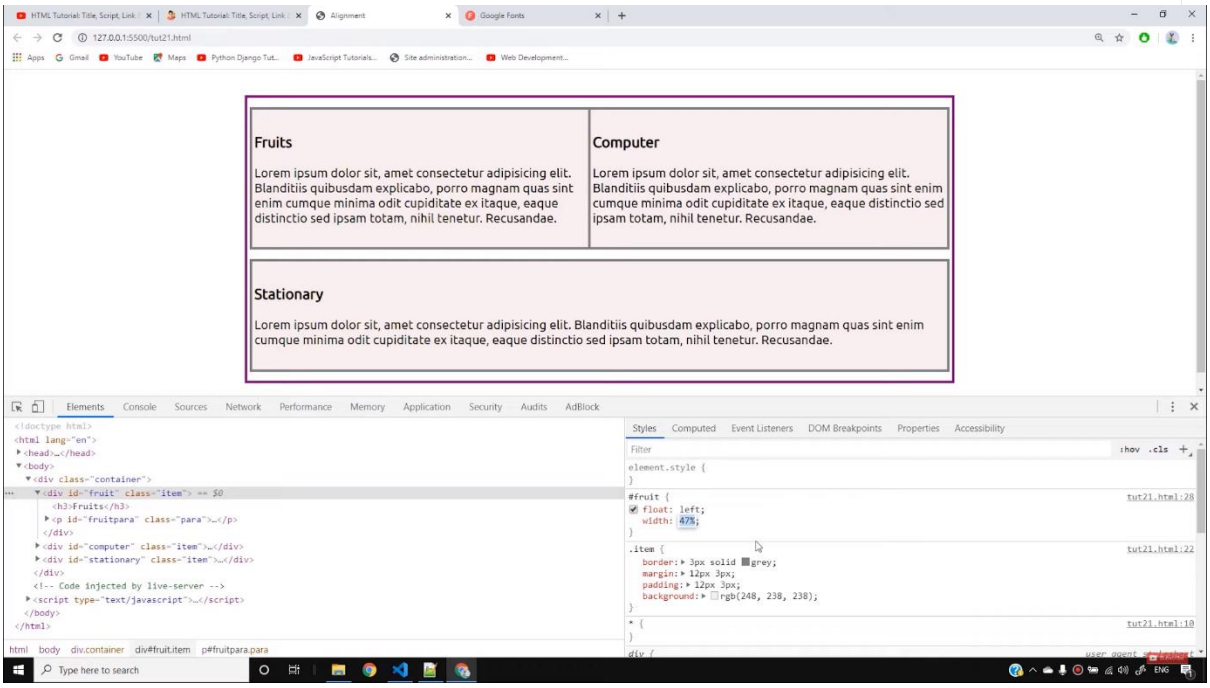


To float the elements, right or left we can target them by their IDs. Let us target all the elements as shown below-

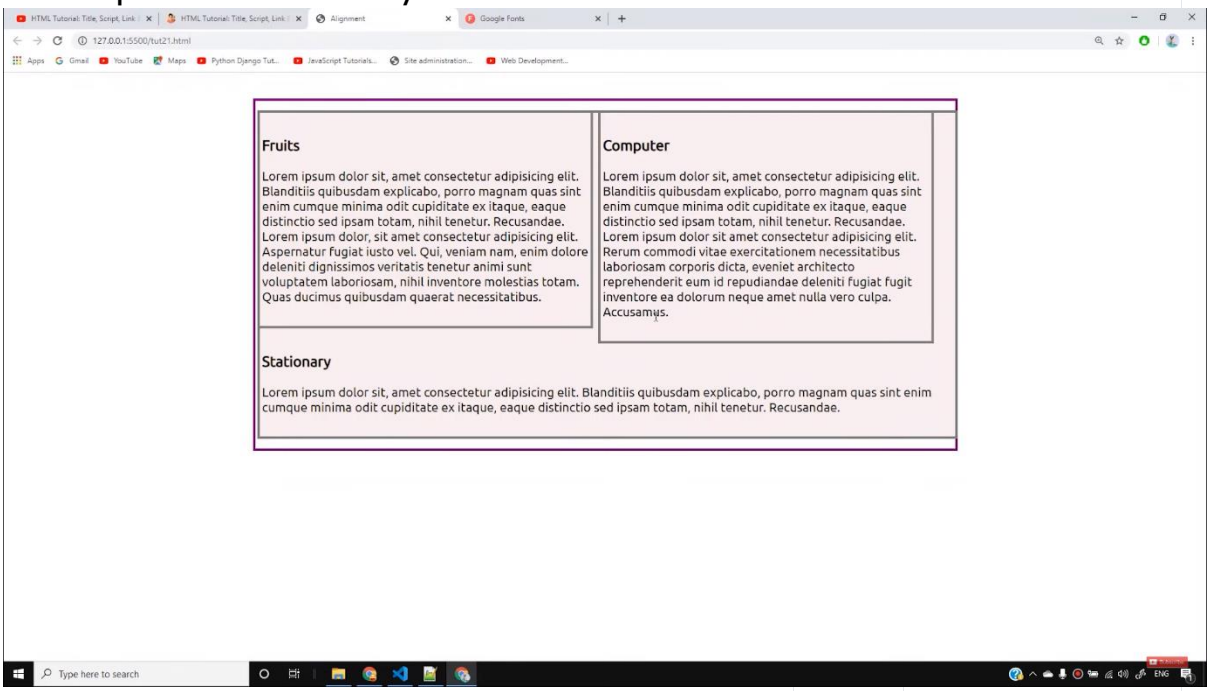
```
#fruit {
            float: right;
            width: 48%;
        }

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

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

Copy

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



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;
        }
```

Copy

However, now we do not use the *float and clear* property much. Instead, we use the properties like *flexbox*. But then too the concepts of float and clear should be known to you. In the

upcoming tutorials, we will see more different layouts like navigation bars etc. Till then stay tuned with the tutorials.

Code as described/written in the video

```
<!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>Alignment</title>
    <link href="https://fonts.googleapis.com/css?family=Ubuntu&display=swap" rel="stylesheet">
    <style>
        * {
            box-sizing: border-box;
        }

        body {
            font-family: 'Ubuntu', sans-serif;
        }

        .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);
        }

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

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

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

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

        h1 {
            margin: 23px auto;
            width: 455px;
        }
    </style>
```

```
</head>

<body>
    <div class="container">
        <h1> Welcome to my store </h1>
        <div id="fruit" class="item">
            <h3>Fruits</h3>
            <p id="fruitpara" class="para">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. Lorem ipsum dolor, sit amet consectetur
                adipisicing elit. Aspernatur fugiat iusto vel. Qui,
veniam nam, enim dolore deleniti dignissimos
                veritatis tenetur animi sunt voluptatem laboriosam,
nihil inventore molestias totam. Quas ducimus
                quibusdam quaerat necessitatibus.</p>
        </div>
        <div id="computer" class="item">
            <h3>Computer</h3>
            <p id="computerpara" class="para">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. Lorem ipsum dolor sit amet consectetur
                adipisicing elit. Rerum commodi vitae exercitationem
necessitatibus laboriosam corporis dicta, eveniet
                architecto reprehenderit eum id repudiandae deleniti
fugiat fugit inventore ea dolorum neque amet nulla
                vero culpa. Accusamus.</p>
        </div>
        <div id="stationary" class="item">
            <h3>Stationary</h3>
            <p id="stationarypara" class="para">Lorem ipsum dolor
sit, amet consectetur adipisicing elit. Blanditiis
```

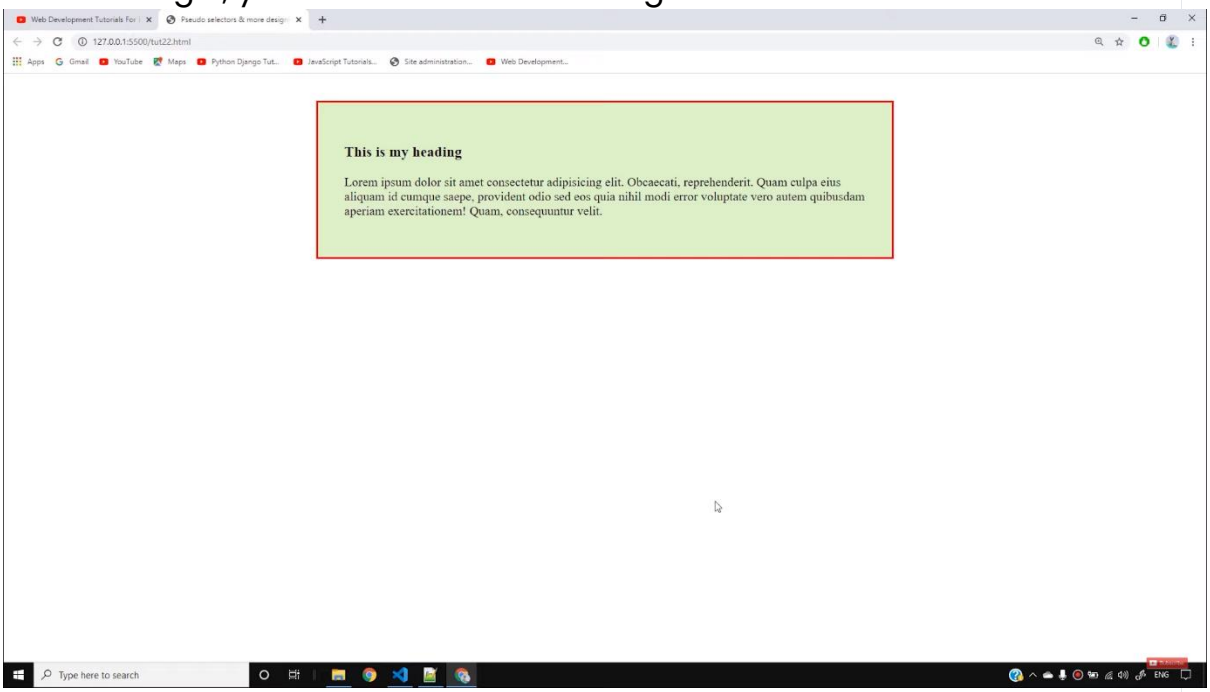
```
                quibusdam explicabo, porro magnam quas sint enim
cumque minima odit cupiditate ex itaque, eaque
                distinctio sed ipsam totam, nihil tenetur.
Recusandae.</p>
        </div>
        <div id="glasses" class="item">
            <h3>Stationary</h3>
            <p id="glassespara" class="para">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.</p>
        </div>
    </div>
</body>


</html>
```

In this tutorial, we are going to see how to **style** and **design** buttons in CSS and what **pseudo-selectors** are. We will start by making a new file as *tut22.html* and then adding the boilerplate to it. Give the title as **Pseudo selectors and more designing** in the <title> tag.
Let us now 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;
        }
```

Copy

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



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>
```

Copy

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.

```
.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;
```

```
        }
```

Copy

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

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

Copy

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

1. **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.

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

Copy

2. 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-

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

Copy

3. 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-

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

Copy

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

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

Copy

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. You can see and practice some of the properties mentioned there and increase your skills. Till then you can visit the previous tutorials and practice all the things taught till now. In the upcoming tutorials, we are going to learn more and more CSS properties and make a website look more attractive from the scratch.

Code as described/written in the video

```
<!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>Pseudo selectors & more designing</title>
```

```
    <style>
        .container{
            border: 2px solid red;
            background-color: rgb(223, 245, 201);
            padding: 34px;
            margin: 34px auto;
            width: 666px;
        }
        a{
            text-decoration: none;
            color: black;
        }
        a:hover{
            color: rgb(5, 0, 0);
            background-color: rgb(221, 166, 38);
        }
        a:visited{
            background-color: yellow;
        }
        a:active{
            background-color:darkblue;
        }
        .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;
        }
        .btn:hover{
            color:darkgoldenrod;
            background-color:rgb(223, 245, 201);
            border: 2px solid black;
        }
    </style>
```

```
</head>
<body>
    <div class="container" id="cont1">
        <h3>This is my heading</h3>
        <p>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.</p>
        <a href="https://yahoo.com" class="btn">Read more</a>
        <button class="btn">Contact us</button>
    </div>
</body>
</html>
```

Give the title as **Navigation** in the <title> tag under the head section.
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>
```

Copy

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;
          }
```

Copy

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;
         }
```

Copy

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;
         }
```

Copy

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;
        }
```

Copy

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>
```

Copy

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;
        }
```

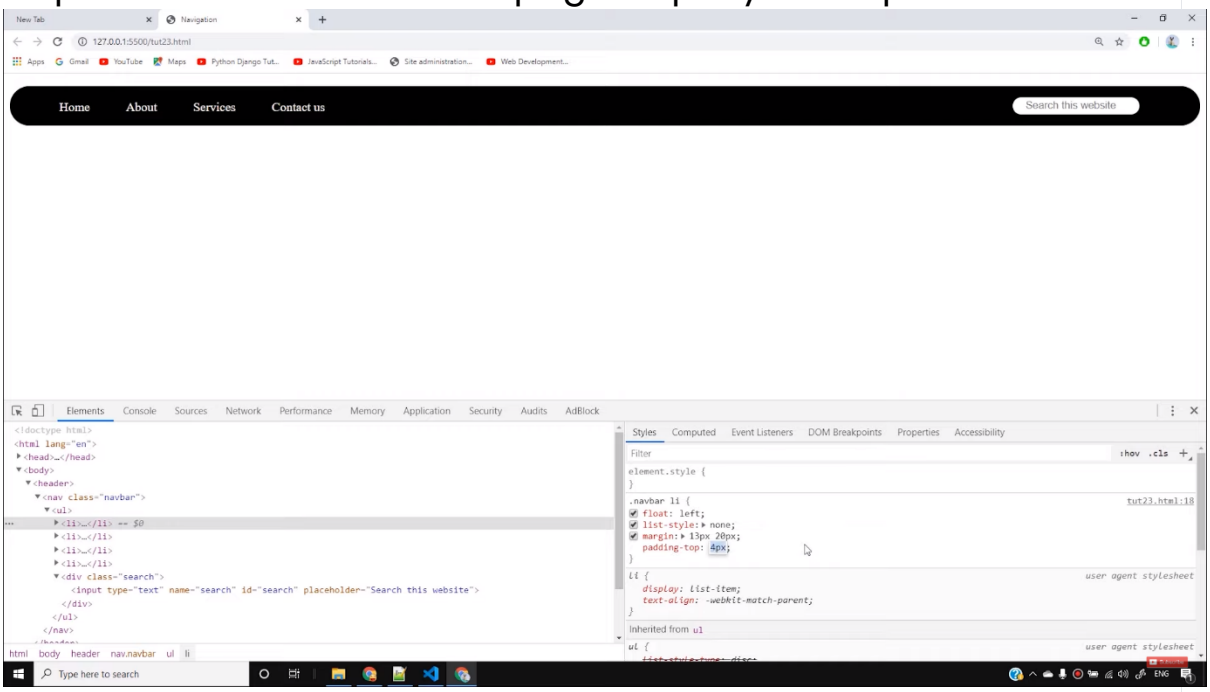
Copy

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;
        }
```

Copy

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
        }
```

Copy

We have set the color to red and now when you hover over the ‘Home’, ‘About’, ‘Services’, ‘Contact-us’, it will change its color to red. You can also add ‘padding-top’ to adjust the elements.
I hope you must have understood how to add the navigation bar into the website and make it look according to yourself. To learn more, stay with the tutorials and keep practicing the things taught till now.

Code as described/written in the video

```
<!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>Navigation</title>
    <style>
        .navbar{
            background-color: black;
            border-radius: 30px;

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

        }
        .navbar li a{
            padding: 3px 3px;
            text-decoration: none;
            color: white;
        }
        .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="#">Contact us</a></li>
                <div class="search">
                    <input type="text" name="search" id="search"
placeholder="Search this website">
                </div>
            </ul>
        </nav>
    </header>
</body>

</html>
```

CSS Display Property in the <title> tag. We will then add an image or logo and the h3 heading, in the header section with the class as “top”.
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;
        }
```

Copy

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-

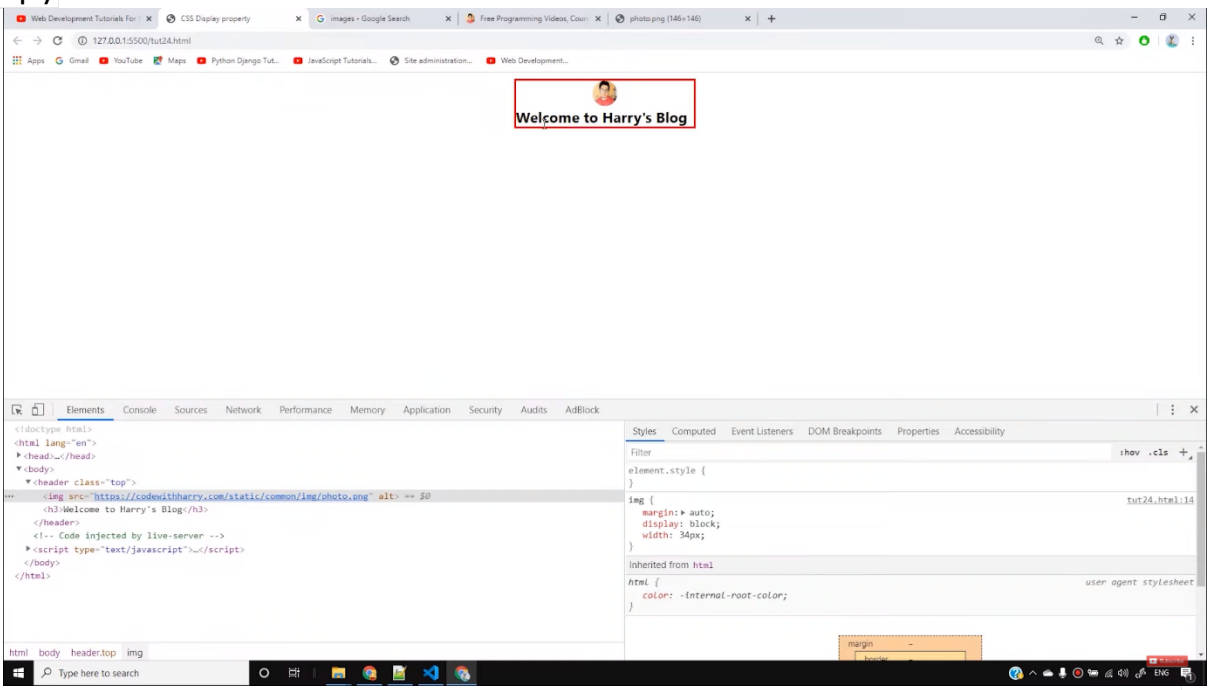
```
 header {
            border: 2px solid red;
            margin: auto;
    width: 1200px;
        }
```

Copy

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;
    }
```

Copy



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.

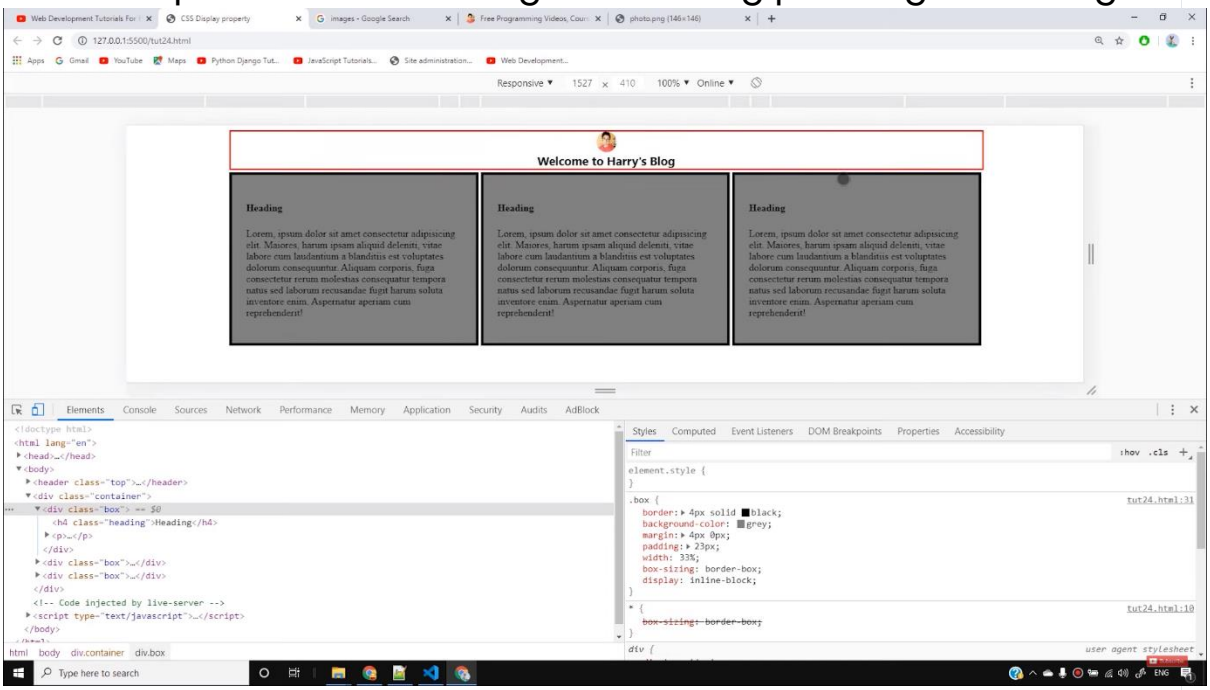
Display inline means it will take the space according to the size of the element. ***Display block*** means we can set its width and by margin manually.

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;
        }
```

Copy

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.



So I believe, you must have clearly understood the concept CSS Display Property. Stay with the tutorials to build more attractive websites in the future. Till then, keep practicing.

Code as described/written in the video

```
<!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="https://codewithharry.com/static/common/img/photo.png" alt="">
        <h3>Welcome to Harry's Blog</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>
```

let's start discussing the CSS positioning-related properties.

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>
```

Copy

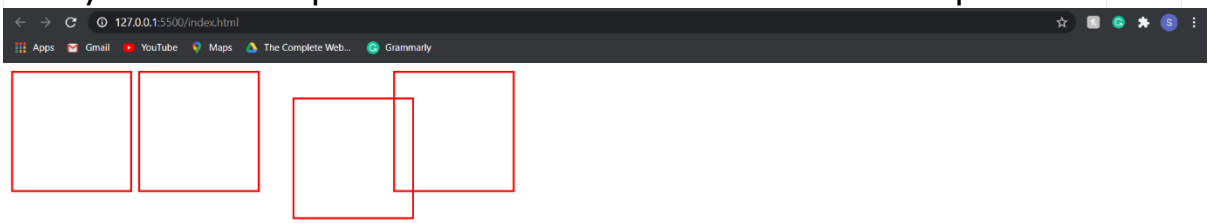
Output:

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>
```

Copy

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.

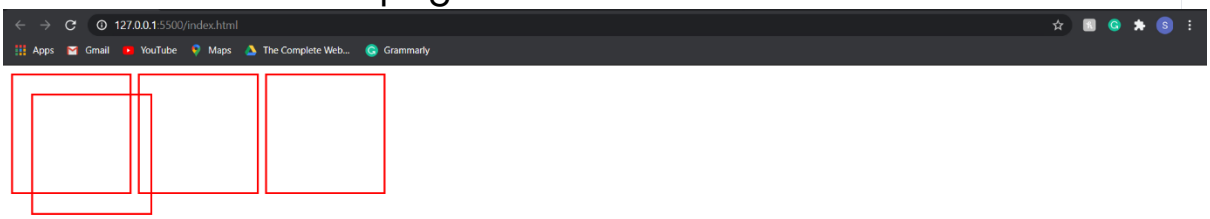
Now, we have changed the position of box3 from relative to absolute. Here is the CSS used :

```
  <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>
```

Copy

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



---

position: fixed;

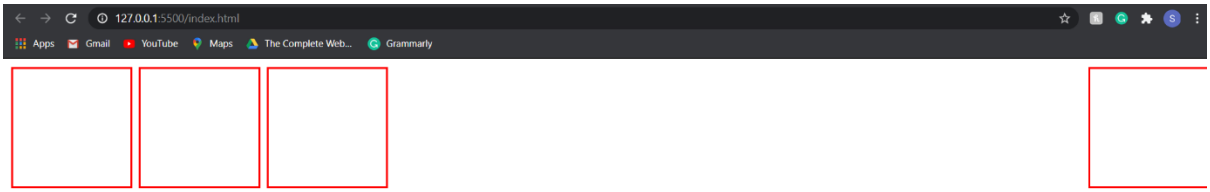
- An element with `position:fixed;` will remain stuck to a specific position even after the page is scrolled.
- This position property is used when we want to keep an HTML element at a fixed spot no matter where on the page the user is.

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>
```

Copy

Output:



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;
            }
```

Copy

We have learnt various properties of CSS, and in the next tutorial, we will design a gym website using the HTML and CSS we have learnt so far. This tutorial ends here, and I will see you in the next tutorial.

Code as described/written in the video

```
<!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>Document</title>
    <style>
        .container{
            border: 2px solid black;
            background-color: khaki;
            height: 3444px;
        }
        /* CSS Position: static (default), relative, absolute, fixed, sticky */
    .box{
        display: inline-block;
        border: 2px solid red;
        width: 150px;
        height: 150px;
        margin: 2px;
    }
    #box3{
        /* relative: Positions the element relative to its normal positon and will leave a gap at its normal position*/
        /* position: relative; */

        /* absolute: Positions the element relative to the positon of its first parent*/
        /* position: absolute; */

        /* top: 34px;
        left: 134px; */

        /* fixed: Positions the element relative to the browser window; */
```

```
        /* position: fixed;
        right: 4px;
        bottom: 2px */

        /* sticky: Positions the element relative to the users
scroll position */
        position: sticky;
        top: 3px;



    }
    </style>
</head>
```

CSS Visibility Property:

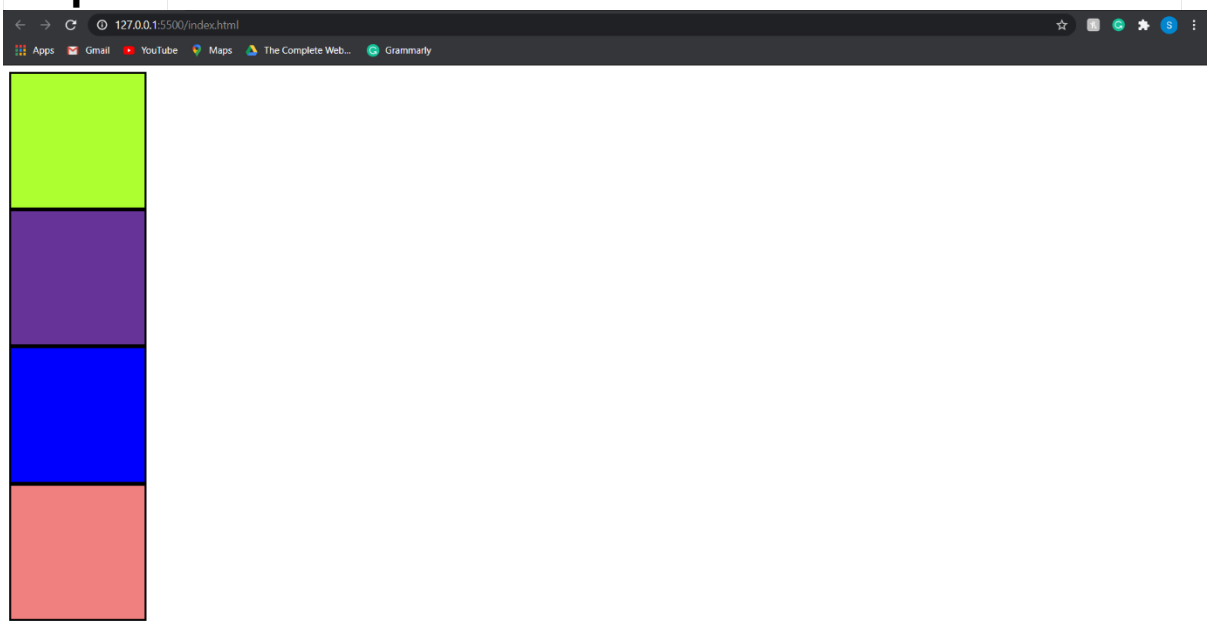
First of all, let's create four boxes of different colours. Here is the CSS used :

```
<!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>
```

Copy

Output :



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

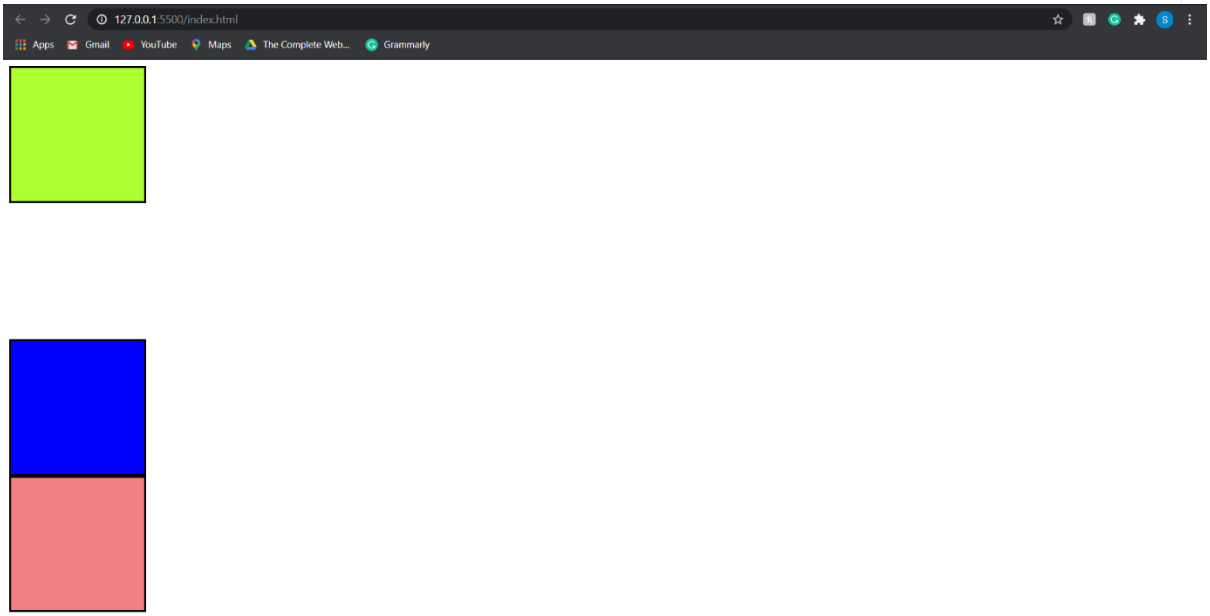
- Visibility property is used to hide or show an HTML element without changing the layout of the page.
- The hidden element uses the space on the page because it is still there, but it is not visible to the user.

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;
        }
```

Copy

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.



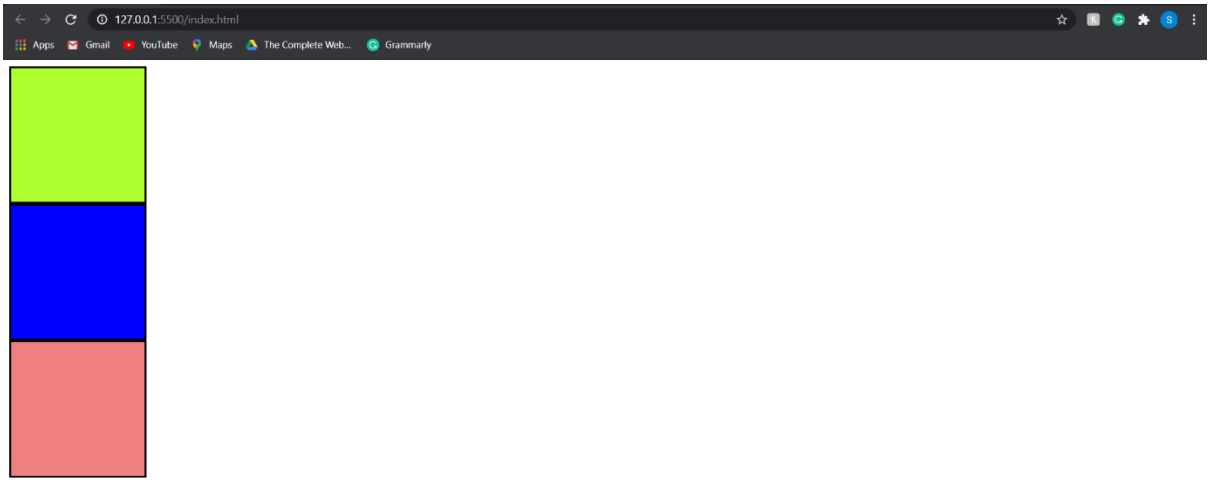
Difference Between `display:none;` and `visibility:hidden;`

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;
        }
```

Copy

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 :

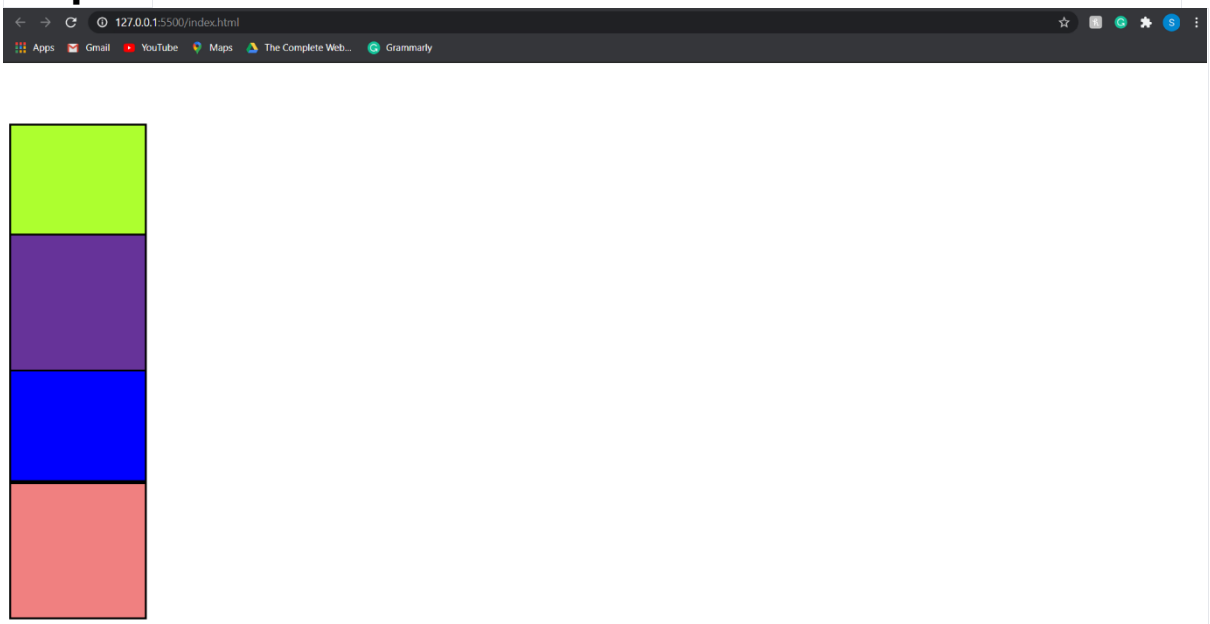
At the starting of this tutorial, we created four boxes of different colours. Now, try to answer this question: What if one box overlaps the other? Which box will be visible to the user? This is where z-index property comes into the picture. So, whenever HTML elements collapse with each other, then the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

Copy

Output:



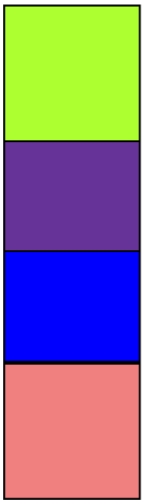
You can see in the above image that the box2 overlaps the box1. Now, we will give z-index value to box1 and box2. Here is the CSS used :

```
        #box1 {
            top: 69px;
            position: relative;
            background-color: greenyellow;
```

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

Copy

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



You can see that the box1 is overlapping the box2 because the z-index value of box1 is greater than the z-index value of box2. So, that's how you can easily change the visibility and z-index value of an HTML element.

This tutorial ends here and I will see in you in the next tutorial.

Code as described/written in the video

```
<!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>Visibility and z-index</title>
    <style>
        .box{
            width: 170px;
            height: 170px;
            border: 2px solid black;
        }
        #box-1{
            position: relative;
            top: 49px;
            z-index: -35;
            background-color: green;        }
        #box-2{
            position: relative;
            top: 14px;
            /* z-index  will work only for position: relative,
absolute, fixed or sticky; */
            z-index: -165;
            /* will hide the element and the space */
            /* display: none;  */
            /* will hide the element but will show its empty space
*/
            /* visibility:hidden;  */
            background-color: red;
            }
        #box-3{
            background-color: blue;        }
        #box-4{ background-color: yellow;        }
    </style>
</head>
<body>
```

```
    <div class="box" id="box-1"></div>
    <div class="box" id="box-2"></div>
    <div class="box" id="box-3"></div>
    <div class="box" id="box-4"></div>
</body>
</html>
```

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. For better understanding watch the complete video.

Code as described/written in the video

```
<!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>Flexbox Tutorial</title>
    <style>
        .container{
            height: 544px;
            width: 100%;
            border: 2px solid black;
            display: flex; /* Initialize the container as a flex box
*/


            /* Flex properties for a flex container */
```

```
            /* flex-direction: row; (Default value of flex-direction is row) */
            /* flex-direction: column;
            flex-direction: row-reverse;
            flex-direction: column-reverse; */


            /* flex-wrap: wrap; (Default value of flex-direction is no-wrap) */
            /* flex-wrap: wrap-reverse; */

            /* This is a shorthand for flex-direction: and flex-wrap: ;; */
            /* flex-flow: row-reverse wrap; */

            /* justify-content will justify the content in horizontal direction */
            /* justify-content: center; */
            /* justify-content: space-between; */
            /* justify-content: space-evenly; */
            /* justify-content: space-around; */

            /* justify-content will justify the content in vertical direction */
            /* 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 */
            /* Higher the order, later it shows up in the container
*/
            /* order: 2; */

            /* flex-grow: 2;
            flex-shrink: 2; */

        }
        #item-2{
            /* flex-grow: 3;
            flex-shrink: 3 ; */
            flex-basis: 160px;
            /* when flex-direction is set to row flex-basis: will
control width */
            /* when flex-direction is set to column flex-basis: will
control height */
        }
        #item-3{
            /* flex: 2 2 230px; */
            align-self: flex-start;
            align-self: flex-end;
            align-self: center;

            }

    </style>
</head>
<body>
    <h1>This is Flexbox Tutorial</h1>
```

```
    <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>
```

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 :

1. By using rem/vh/vw units over pixels.

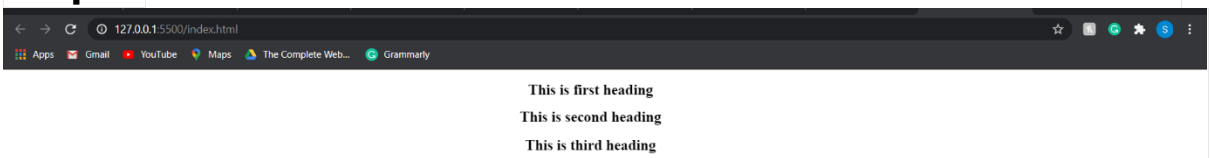
2. By using max-width/max-height.
3. Using CSS Media Queries.
4. Setting up the viewport.

This tutorial will cover the use of CSS size units to make a responsive website. We will see other ways of responsive design in further tutorials. So, let's start our discussion on CSS size units. First of all, we are using the below CSS to create three headings:

```
<!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>
```

Copy

Output:



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.

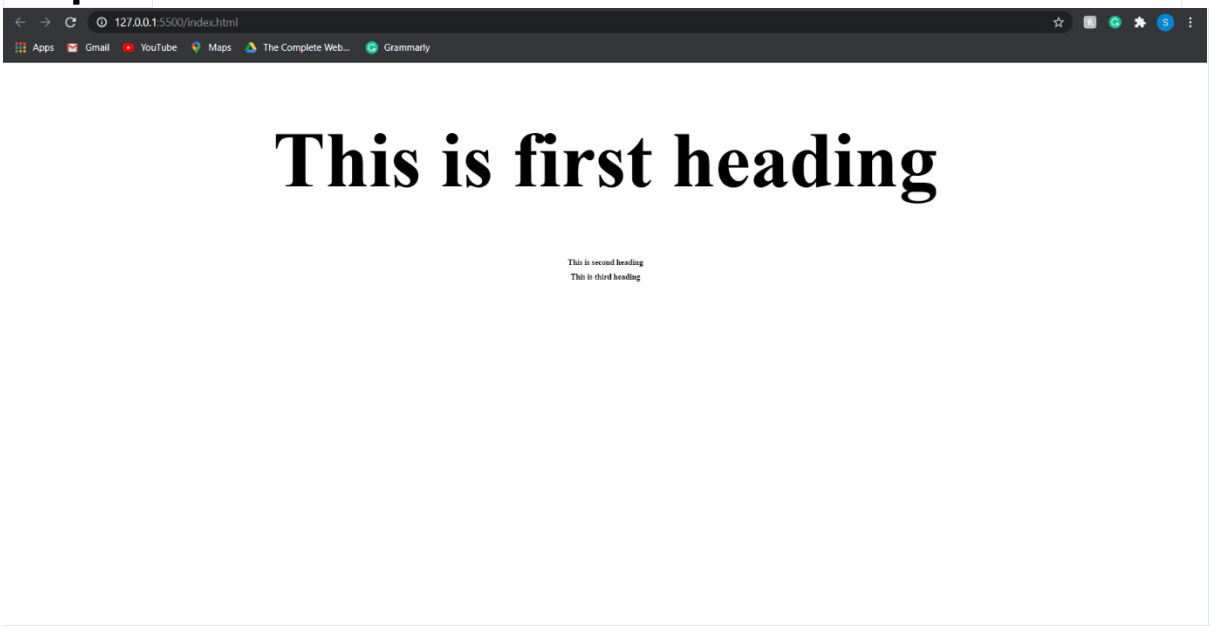
Use the CSS given below :

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

```
</style>
```

Copy

Output:



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."

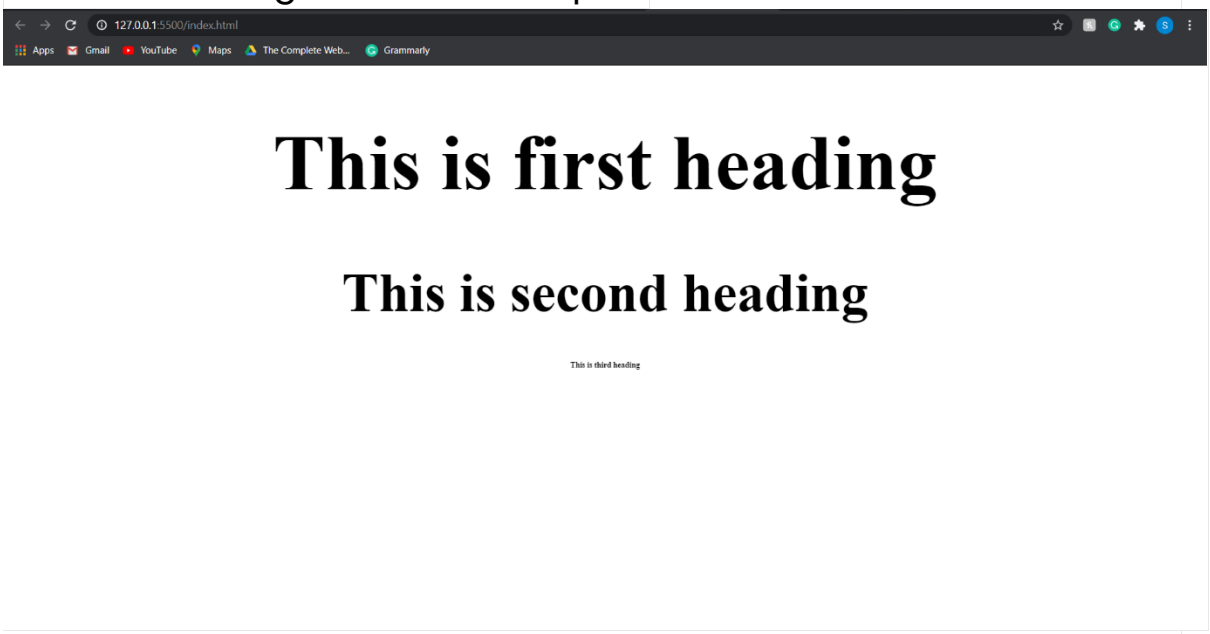
Use the CSS given below:

```
<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>
```

Copy

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 :



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-

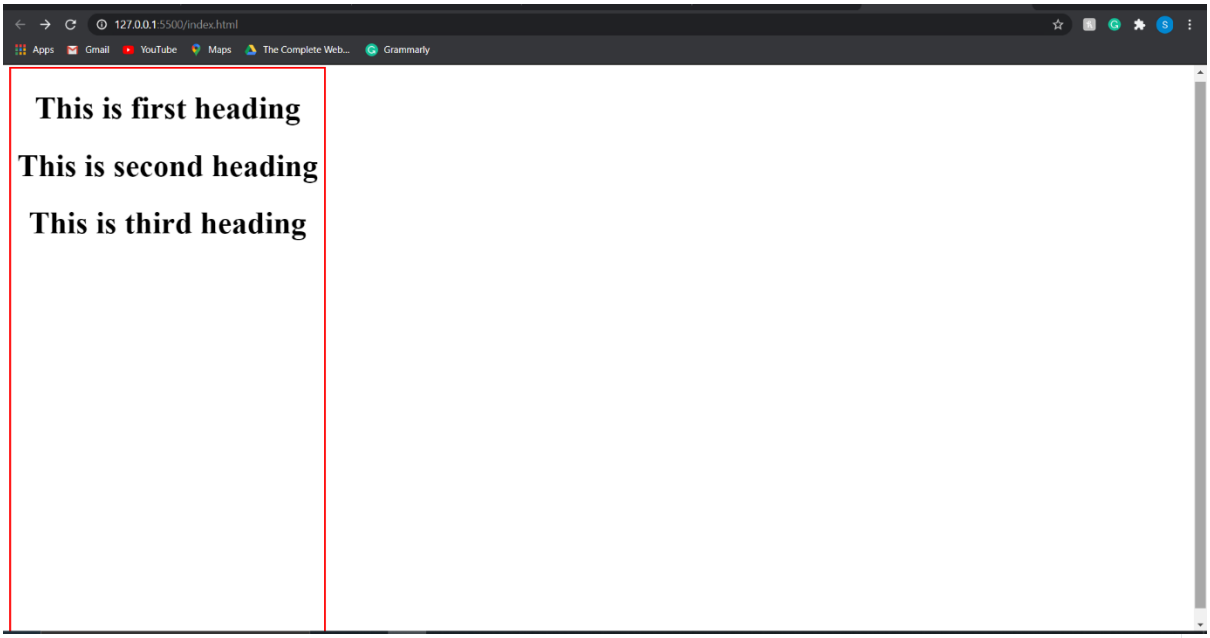
- It stands for viewport height.
- vh is equal to the 1/100 times of the viewport height.
- Example: Suppose height of the browser is 1000px, so 1vh is equaled to (1/100)*1000= 10px.

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>
```

Copy

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



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.

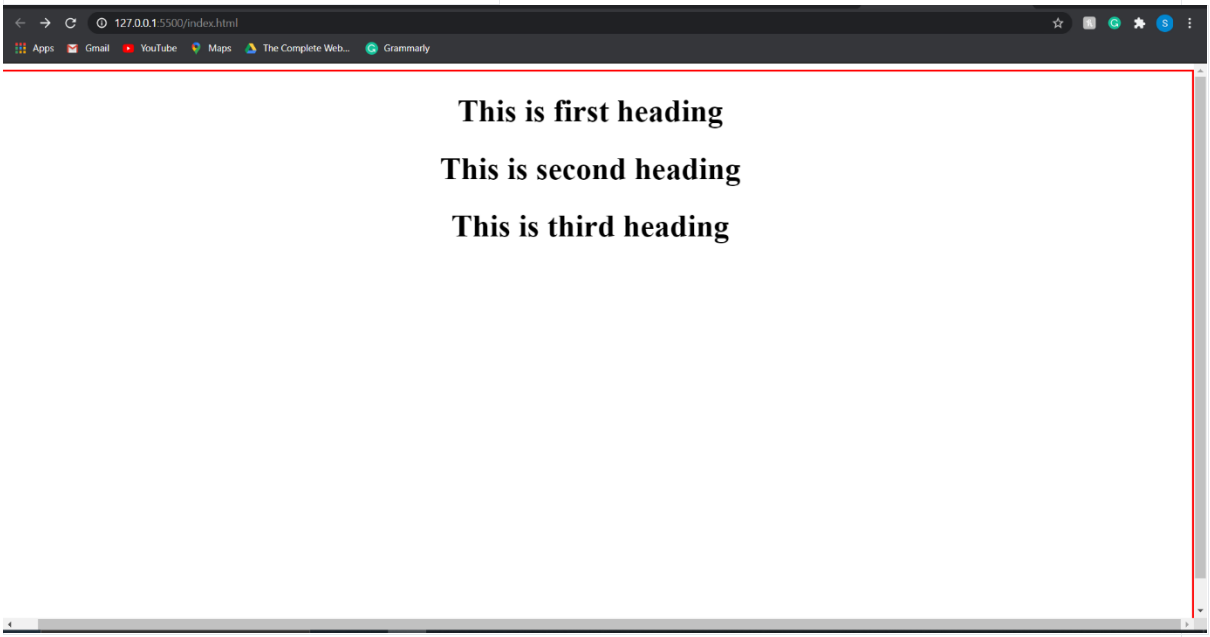
Use the CSS given below:

```
<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>
```

Copy

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



In the above image, you can see that the border's width is changed to 100% of the viewport. This tutorial ends here, and I believe that all the concepts discussed in this tutorial are crystal clear to you.

Code as described/written in the video

```
<!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>Size units</title>
    <style>
        html{
            font-size: 25px;
        }
        .container{
            width:400px;
            /* height: 344px; */
            height: 100vh;
            width: 100vw;
            font-size: 10px;
            border :2px solid red;
        }
    h1{
        text-align: center;
    }
    #first{
        /* font-size: 3em;
        padding: 3em; */

    }
    #second{
        /* font-size: 3rem;
        padding: 3rem; */

    }

    </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>
```

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*/
}
```

Copy

Example:

```
@media screen and (max-width: 800px)
{
```

```
#contents{width:90%}
}
```

Copy

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>
```

Copy

Here is the output :

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>
```

Copy

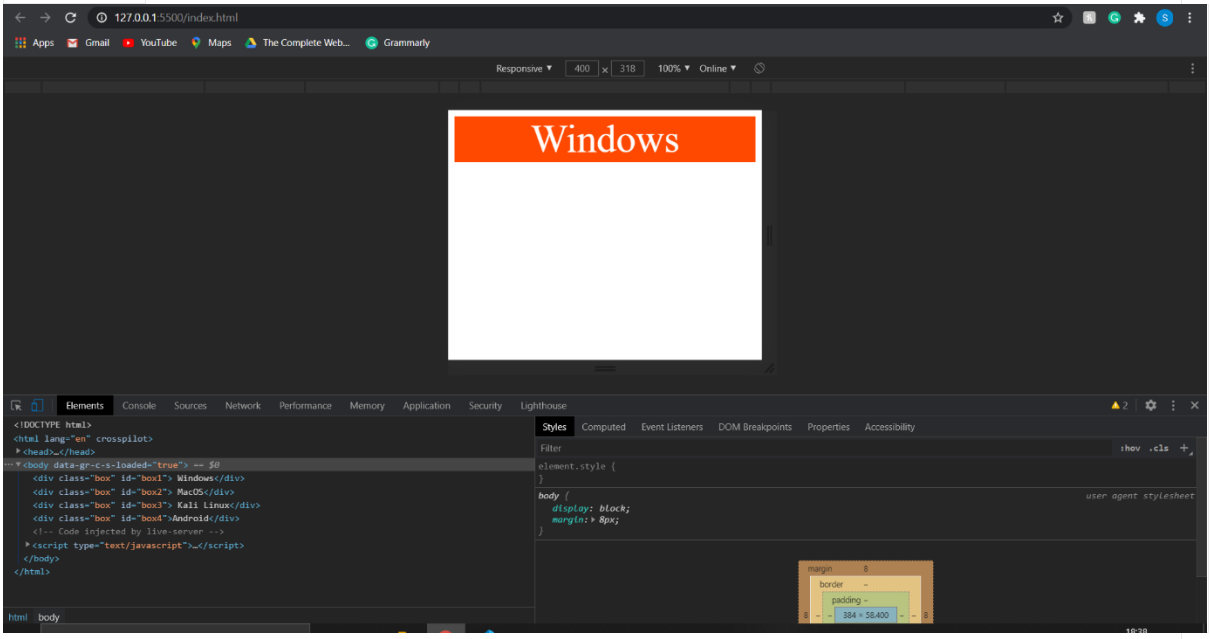
Now, it's time to apply our very first media query. Here is the CSS used :

```
@media (min-width: 400px){
            #box1{
                display: block;
            }
        }
```

Copy

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

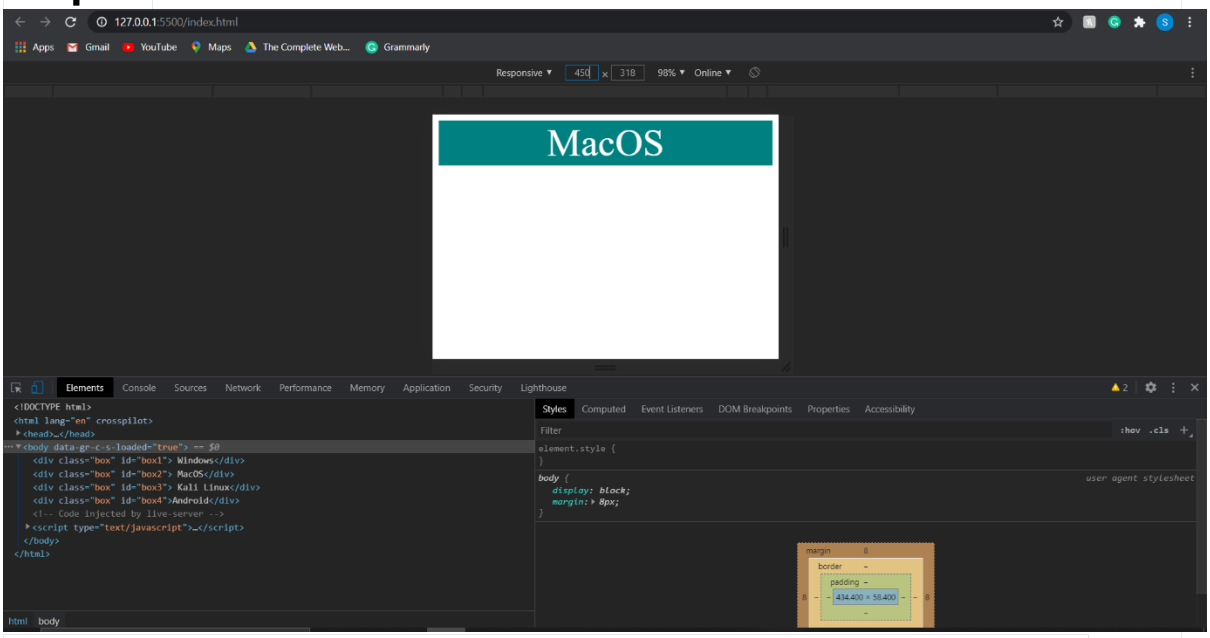
width to 400px. Similarly, we have applied a media query for the box2. Here is the CSS used:

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

Copy

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 :



So, that's how you can easily include different media queries depending upon your requirements. Note that we have

used **and** operator to combine two queries to be true. You can also use other operators such as not, only, and comma(,). So, this is all for this tutorial, and I hope you have got the basic knowledge of media queries in CSS. Feel free to ask your queries in the QnA section.

Code as described/written in the video

```
<!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>Media Queries</title>
    <style>
        .box{
            font-size: 72px;
            text-align: center;
            background-color: red;
            color: white;
            display: none;
        }

        @media only screen and (max-width:300px){
            #box-1{
                display: block;
                background-color: cyan;
            }
        }

        @media only screen and (min-width: 300px) and (max-width:500px) {
            #box-2{
```

```
                display: block;
                background-color: blueviolet;
            }
        }

        @media (min-width: 500px) and (max-width:800px) {
            #box-3{
                display: block;
                color: black;
                background-color: yellow;
            }
        }

        @media (min-width: 800px)  {
            #box-4{
                display: block;
                background-color: green;
            }
        }

    </style>
</head>
<body>
    <div class="box" id="box-1">Mai ek iPhone hoon</div>
    <div class="box" id="box-2">Mai ek Tablet hoon</div>
    <div class="box" id="box-3">Mai ek desktop computer hoon</div>
    <div class="box" id="box-4">Mai ek widescreen computer
hoon</div>
</body>
</html>
```

advanced concepts about CSS selectors.

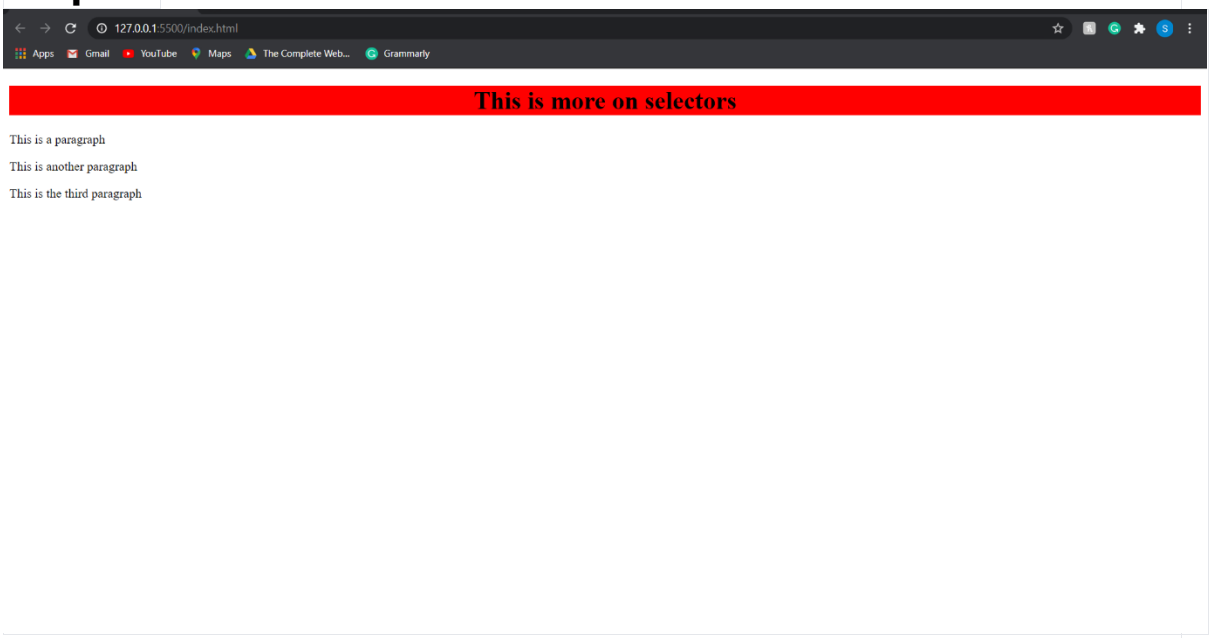
First of all, we have used the CSS given below to design some divs and paragraph tags.

```
<!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>
```

Copy

Output :

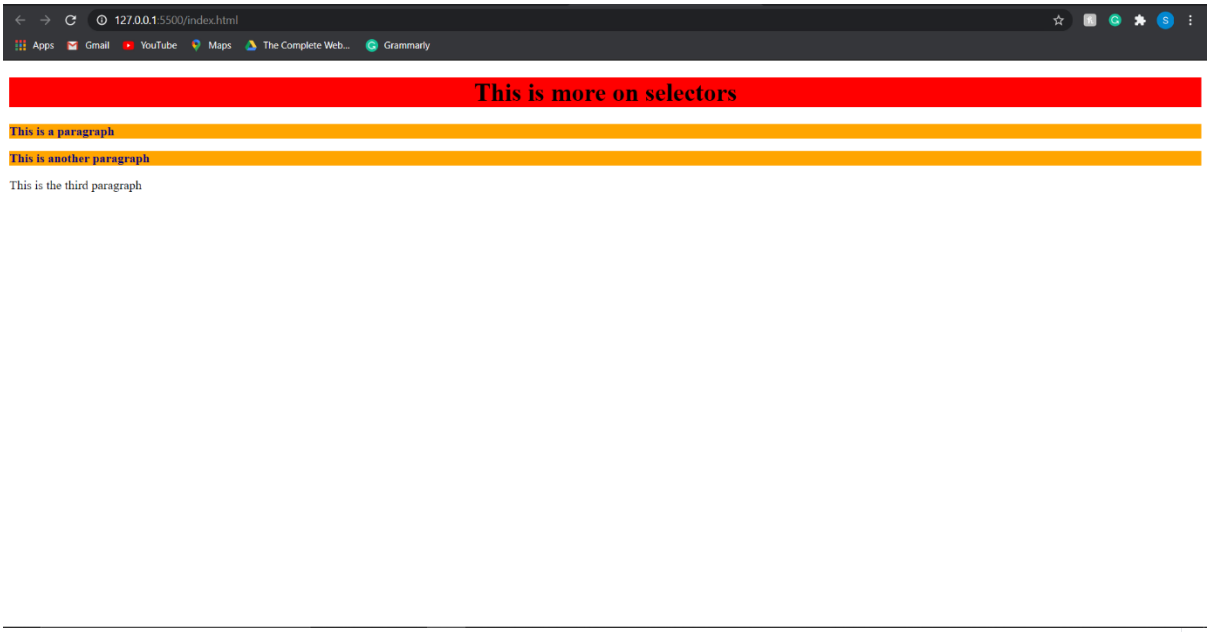


Now, let's suppose you want to add styling to all the paragraphs inside the div. What will you do? You can use the following CSS :

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

Copy

Here are the results :



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;
```

Copy

Example :

```
div>p
```

Copy

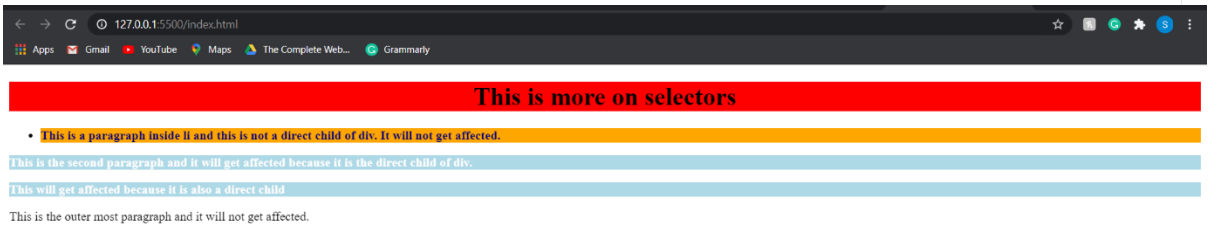
In the above example, the styling will be applied to all <p> elements which are the direct child of any div element. Let's understand this with the help of paragraph tags that we created at the starting of this tutorial. Here is the CSS used :

```
<!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;
        }
        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>
```

Copy

Here are the results :



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 :

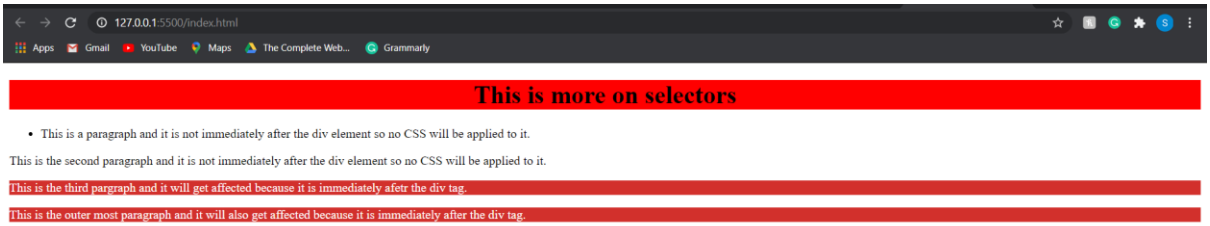
There might be situations where you want to select the <p> tags that are immediately after the elements. In such cases, we use the div+p selectors. Let's understand this with the help of example given below:
Html and CSS used:

```
<!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>
```

Copy

Output :



In the above image, you can see that the CSS is applied only to the third and outermost paragraphs because they are next to the element.
This is all for this tutorial, and in the next tutorial, we will discuss the nth-child pseudo selectors. See you in the next tutorial!

Code as described/written in the video

```
<!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>More on selectors</title>
</head>
<style>
    h1{
        background-color: red;
        color: black;
        font-weight: bold;
        text-align: center;
```

```
    }

    /* if p is contained by any li which is contained by div */
    /* div li p{
        color: yellow;
        background-color: green;
        font-weight: bold;
    } */

    /* if p is right inside div then this CSS will be applied */
    /* div > p{
        color: yellow;
        background-color: green;
        font-weight: bold;
    } */

    /* if p is right after div i.e p is the next sibling of div*/
    /* div + p{
        color: white;
        background-color: rgb(238, 137, 137);
    } */

</style>
<body>
    <h1>This is more on selectors</h1>
    <div class="container">
        <div class="row">
            <ul>
                <li class="item"><p> this is another paragraph
inside li</p></li>
                <li>this will not get affected</li>
                <p>this is a para inside ul</p>
            </ul>
            <p>This is a paragraph</p>
```

```
        </div>
        <p>This is another paragraph</p>
    </div>
    <p>this is outermost paragraph</p>
</body>
</html>
```

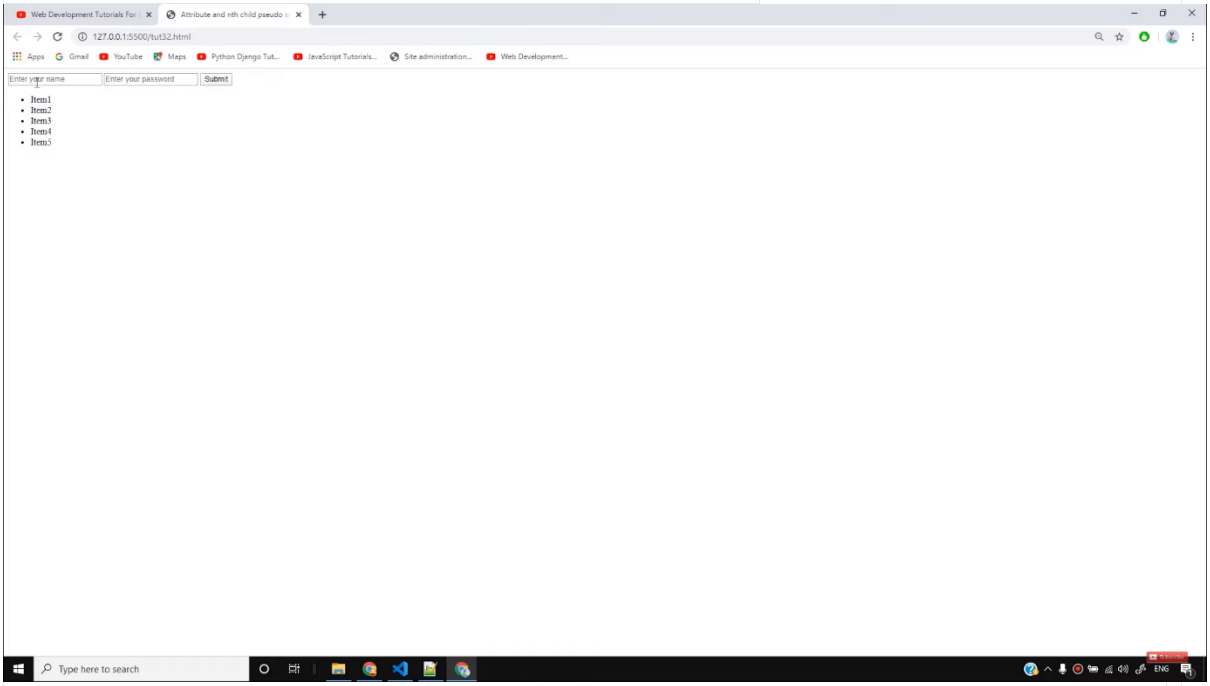
Give the title as an ***attribute and nth-child pseudo-selectors*** in the <title> tag.
We will create a very basic form by writing the HTML code as below-

```
<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>
```

Copy

The form will look very simple as follows :



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

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

Copy

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;
        }
```

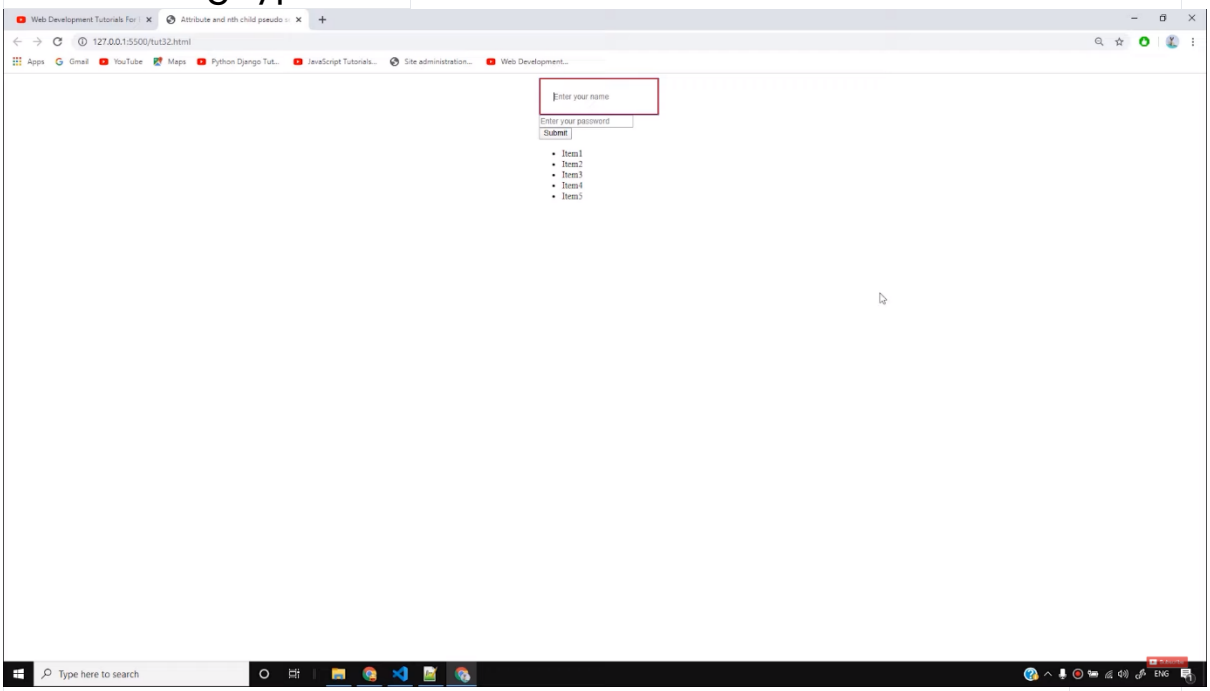
Copy

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

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

Copy

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;
        }
```

Copy

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;
}
```

Copy

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;
        }
```

Copy

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;
```

```
        }
```

Copy

You can try more yourself by practicing with different codes and analyze the changes accordingly. To keep learning more, stay with the tutorial.

Code as described/written in the video

```
<!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>Attribute and nth child pseudo selectors</title>
    <style>
        .container {
            display: block;
            width: 233px;
            margin: auto;
        }

        input {
            display: block;
        }

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

        a[target] {
```

```
            font-size: 44px;
            color: violet;
        }

        a[target='_self'] {
            font-size: 14px;
            color: rgb(13, 22, 151);
        }

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

        /* This will apply css for third child */
        /* li:nth-child(3){
        color: cyan;
    }

    li:nth-child(2n+0){
        color: red;
    } */

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

        /* Odd child */
        /* li:nth-child(odd){
        background-color: yellow;
    } */

        /* Even child */
        li:nth-child(even) {
```

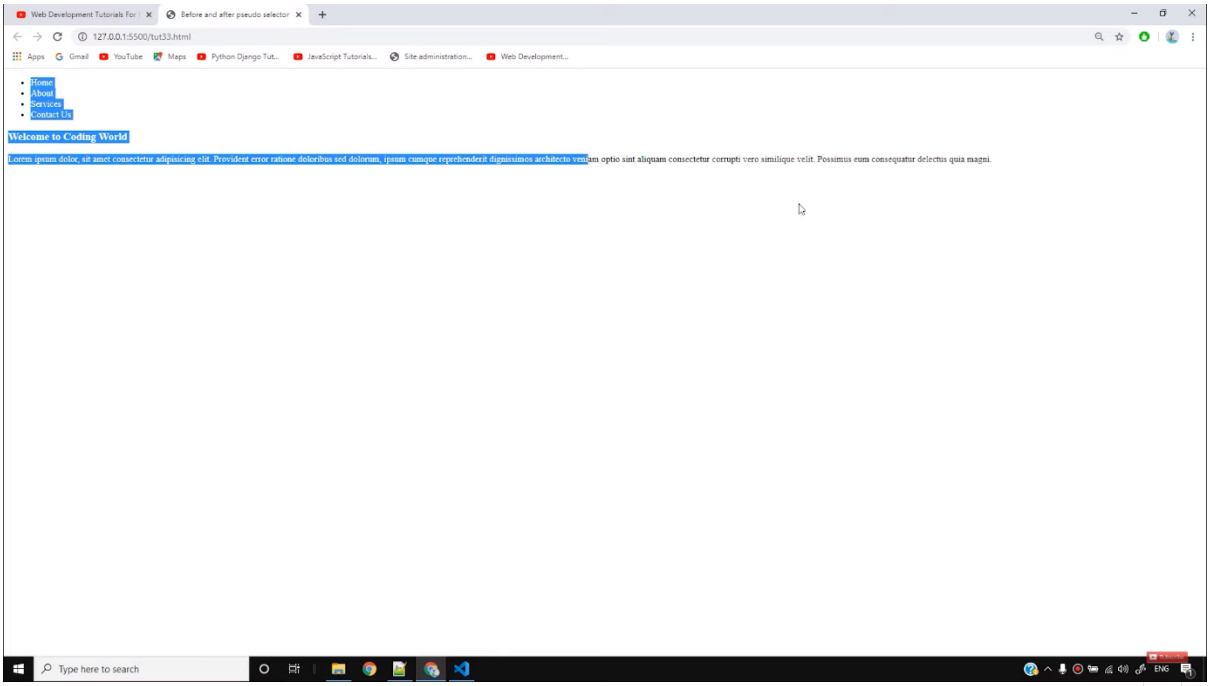
```
            background-color: yellow;
        }
    </style>
</head>

<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>

</html>
```

the title as **before and after pseudo-selectors** under the <title> tag.
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>
```

Copy

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



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

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

Copy

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;
        }
```

Copy

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

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

Copy

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;
        }
```

Copy

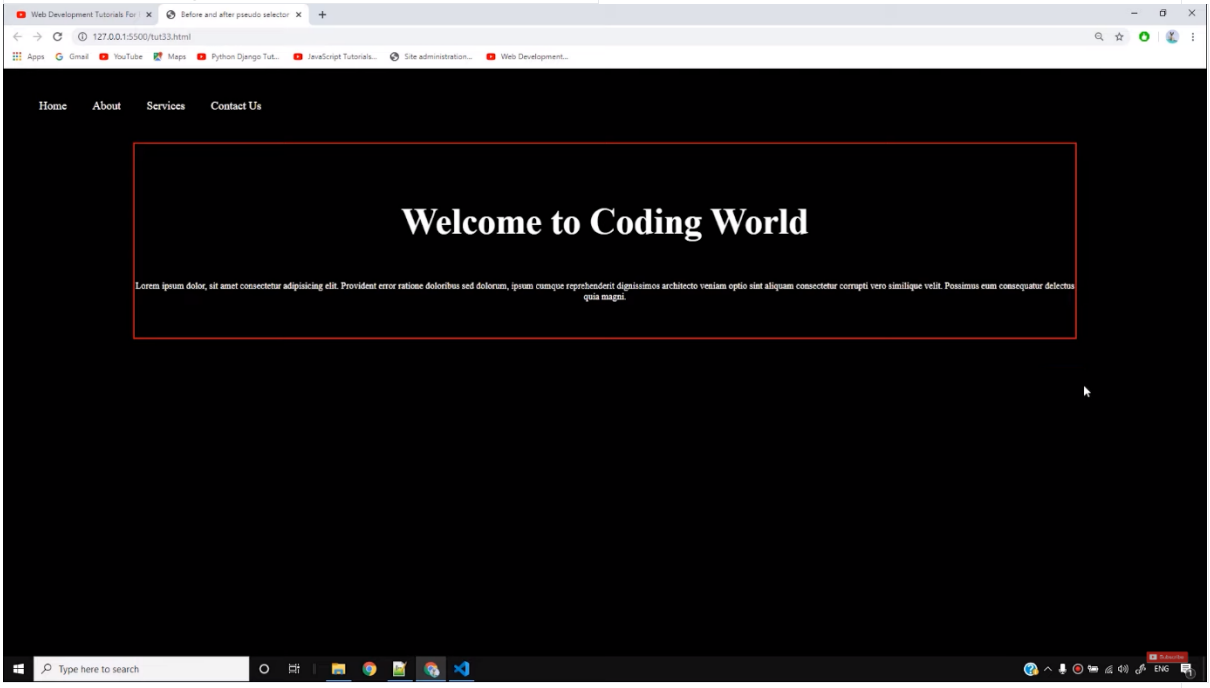
Then increase the font size of the heading by **4 rems** and take the content of the paragraph in the **center** as follows-

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

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

Copy

So our page will look as follows-



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.

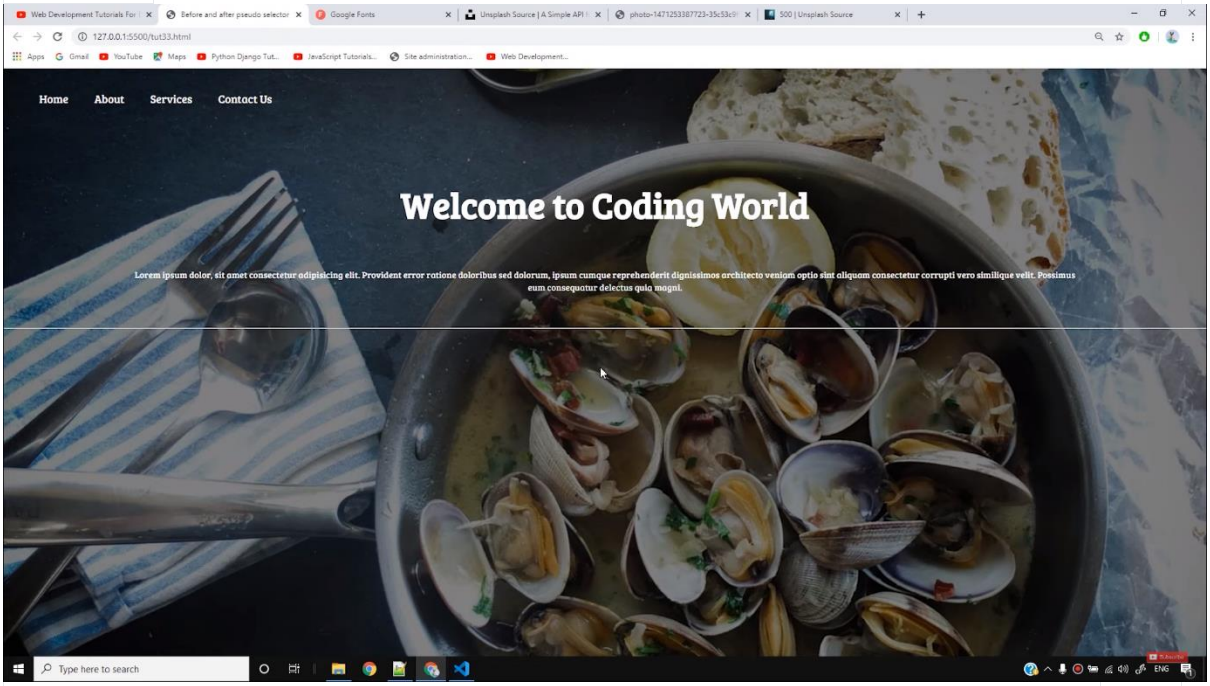
To adjust this problem, we can add content before the header. In that content, we can add a background image, makes its position as **absolute,** and set its width and height as follows-

```
  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;
        }
```

Copy

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.** So our final website will look as follows-



In this tutorial, we have learned how with the help of before and after pseudo selectors, we can change the opacity of the

background image. For learning more about different techniques, stay tunes with the tutorials.

Code as described/written in the video

```
<!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>Before and after pseudo selector</title>
    <link href="https://fonts.googleapis.com/css?family=Bree+Serif&display=swap" rel="stylesheet">
    <style>
        body {
            margin: 0;
            padding: 0;
            background-color: black;
            color: white;
        }

        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;
        }

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

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

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

        h1 {
            font-size: 4rem;
        }

        p {
            text-align: center;
        }

        /* section::after{
```

```
        content:"this is a content"
    } */

    </style>
</head>

<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>

</html>
```

Box Shadow and **Text Shadow.** As usual, make a new file as *tut34.html* and then add the basic boilerplate. Give the title as *Box Shadow and Text Shadow* in the <title> tag.
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>
```

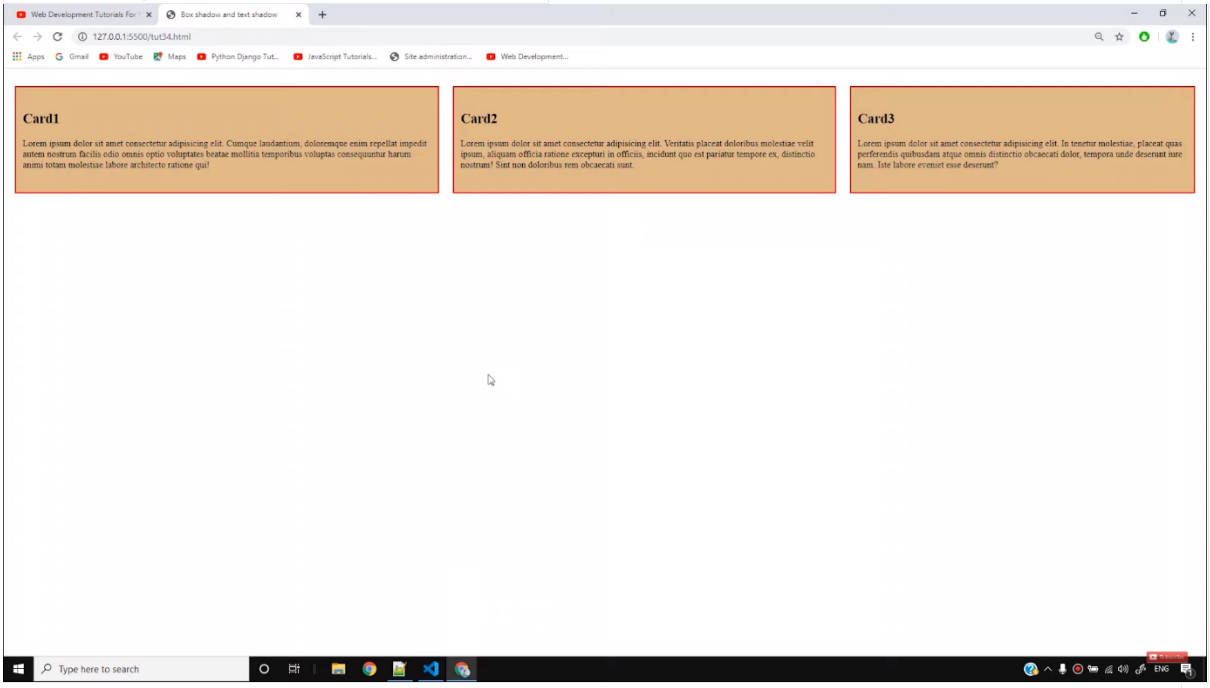
Copy

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;
}
```

Copy

Our page will look as follows-

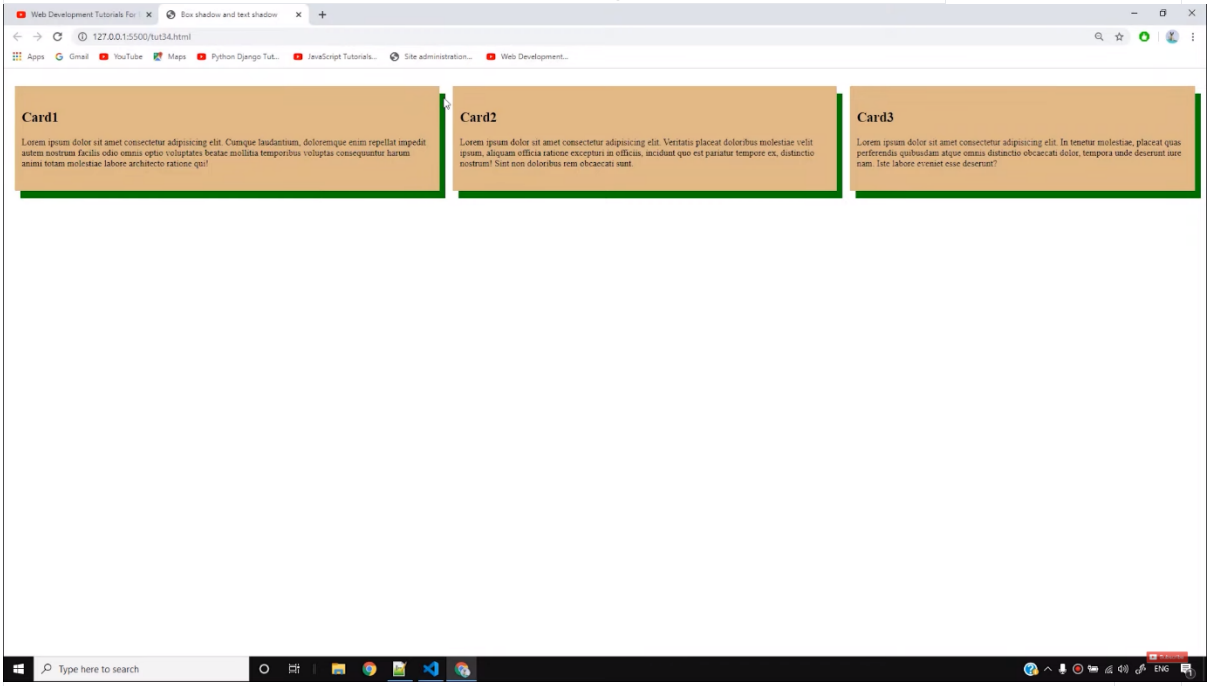


Now let us understand how to apply the box-shadow effect in the CSS. The basic syntax is as-

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

Copy

Now the boxes inside the webpage will appear as-



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

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

Copy

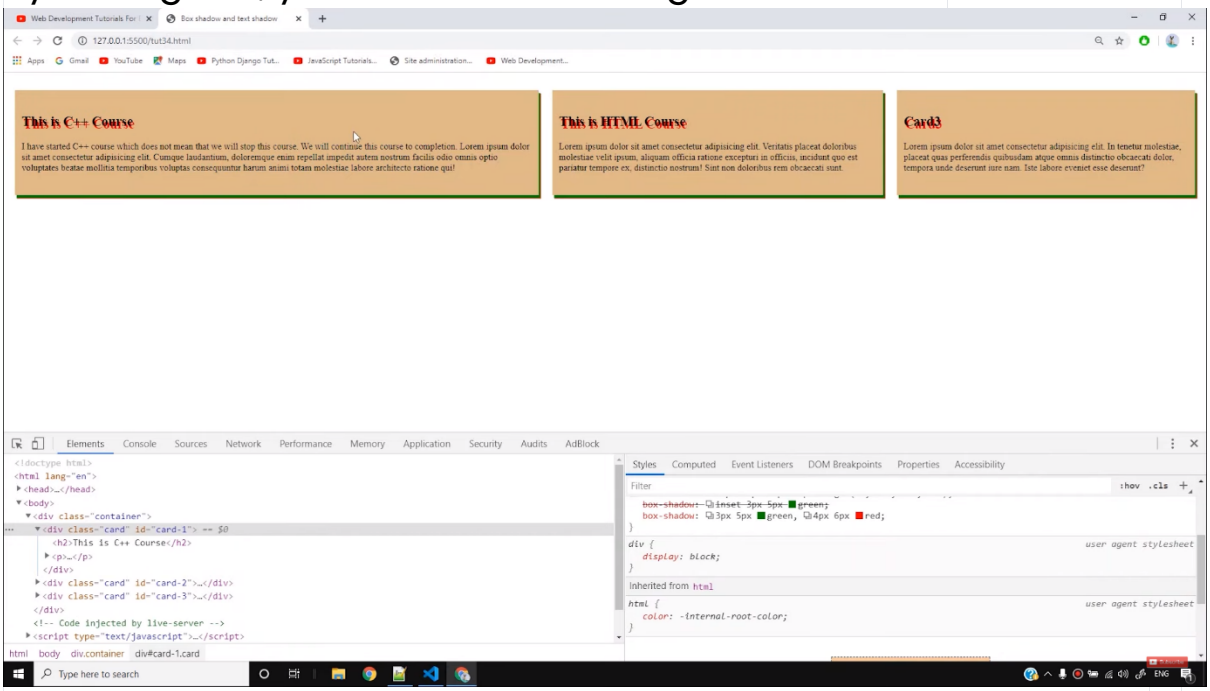
Now let us see how to use **text-shadow.** For this, we will make the changes in the heading tag as follows-

```
.card h2{
```

```
        text-shadow: 3px 4px red;
}
```

Copy

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



We can use the negative values here also to see the changes towards the upside. All the properties like blur-radius and spread-radius can also be used here.

So, I believe you have understood the concepts of box and text shadows. You can try different effects and analyze the changes accordingly. Till then, stay connected with the tutorial.

Code as described/written in the video

```
<!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>Box shadow and text shadow</title>
</head>
<style>
.container{
    display: flex;
}
.card{
    padding: 23px 12px;
    margin: 23px 12px;
    /* border: 2px solid red; */
    background-color: burlywood;
    /* box-shadow: offset-x offset-y color; */
    /* box-shadow: offset-x offset-y blur-radius color; */
    /* box-shadow: offset-x offset-y blur-radius spread-radius
color; */

    /* box-shadow: 10px 13px green; */
    /* box-shadow: -10px -13px green; */
    /* box-shadow: 7px 5px 10px green;
    box-shadow: -7px -5px 10px green; */
    /* box-shadow: -7px -5px 10px 34px green; */
    /* box-shadow: -7px -5px 10px 34px rgba(71, 172, 172, 0.5); */
    box-shadow: inset 3px 5px green;

    box-shadow: 3px 5px green, 4px 6px red;
}
.card h2{
    /* text-shadow: 3px 4px red; */
    /* text-shadow: 3px 2px 2px white; */
    text-shadow: -3px -2px 2px white;
}
</style>
<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>
</html>
```

CSS variables and custom properties in the <title> tag.
Now don't get confused and start comparing these variables with the other programming languages. 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>
```

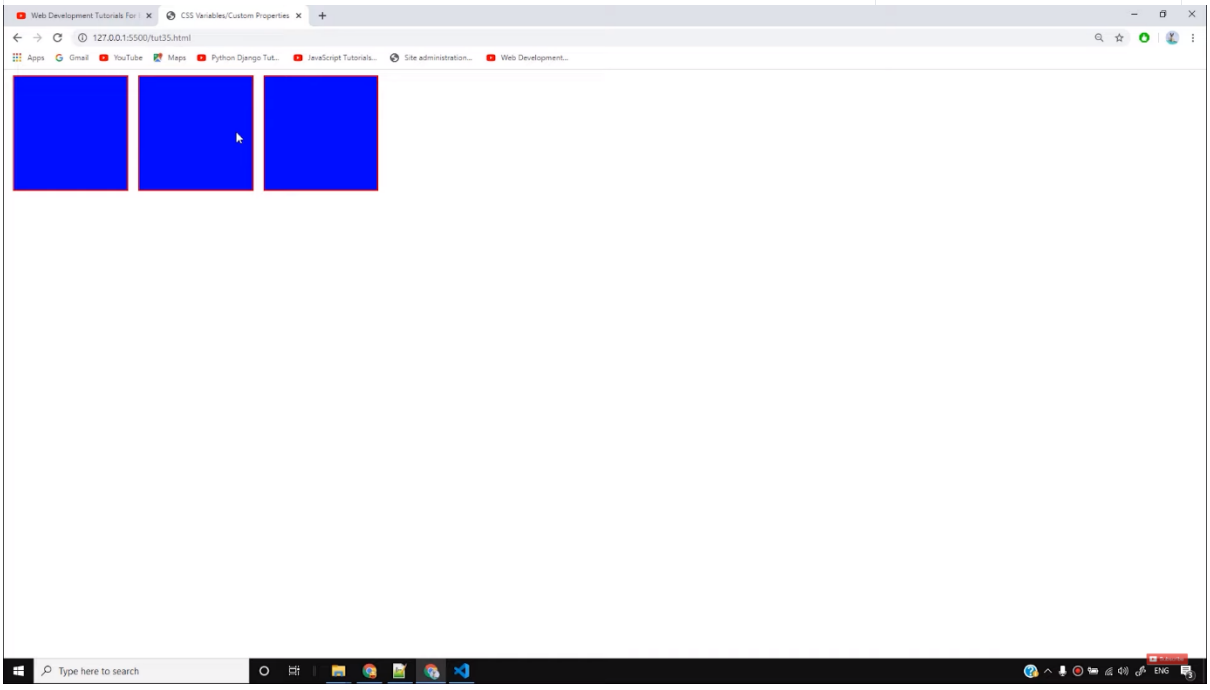
Copy

Let us style the box as follows-

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

Copy

The result of the above code will be as follows-

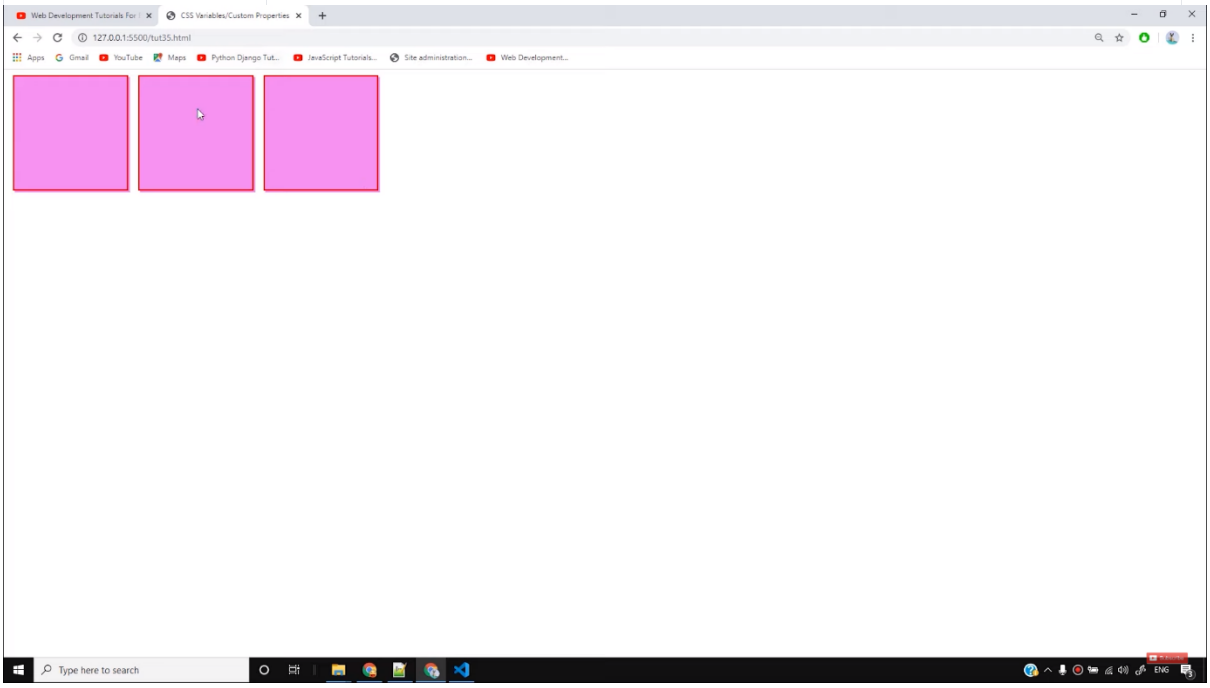


Now we will understand the concept of **variables.** Suppose, we want to create a variable for the background color. We can create it by **'--'** symbol. Variables in CSS helps us to assign the same properties to different elements. Let us analyze it with the code given below-

```
.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;
        }
```

Copy

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;
        }
```

Copy

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;
        }
```

Copy

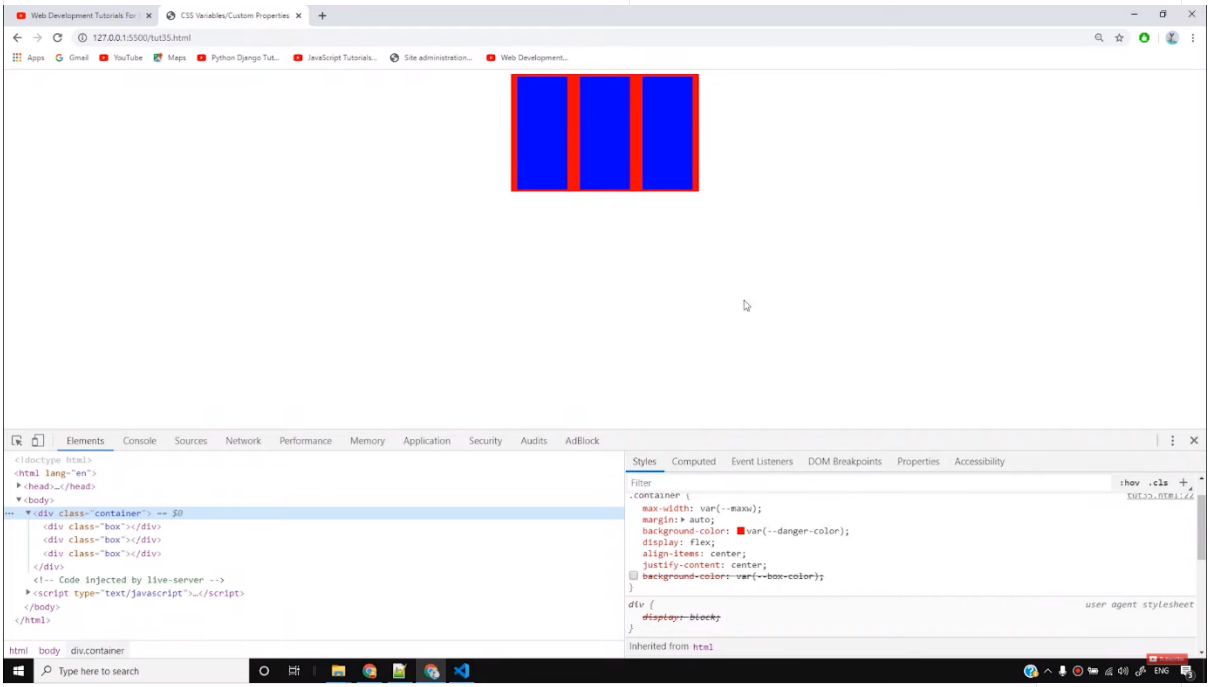
Now let us modify the container and add the **maxw** property in it from the root with additional some properties. The code is as follows-

```
 .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); */
        }
```

Copy

The container will look as follows-



So, I believe you must have understood the concepts of variables and custom properties in CSS and how they can be used to minimize our efforts. For more interesting upcoming tutorials, kindly stay tuned and keep practicing.

Code as described/written in the video

```
<!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 Variables/Custom Properties</title>
    <style>
        :root{
            --primary-color: blue;
            --danger-color: red;
            --maxw: 333px;
        }
        .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;
        }
        .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); */
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box"></div>
        <div class="box"></div>
        <div class="box"></div>
```

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

Give the title as **CSS variables and custom properties** in the <title> tag.
Now don't get confused and start comparing these variables with the other programming languages. 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>
```

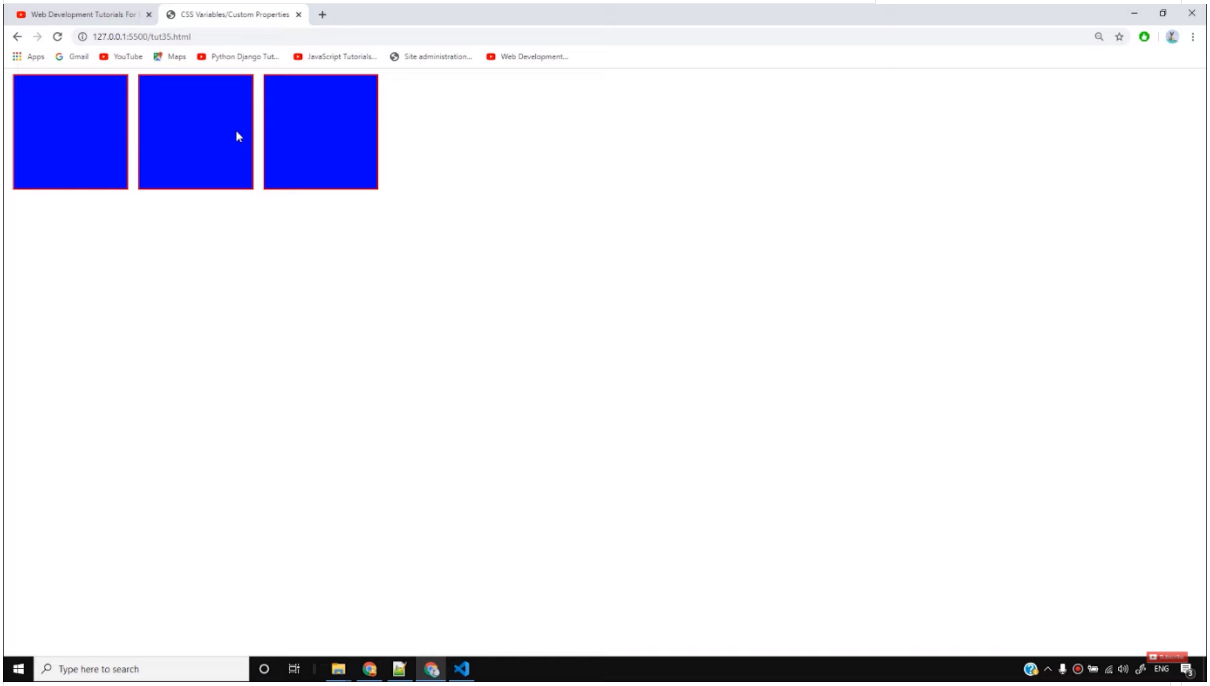
Copy

Let us style the box as follows-

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

Copy

The result of the above code will be as follows-

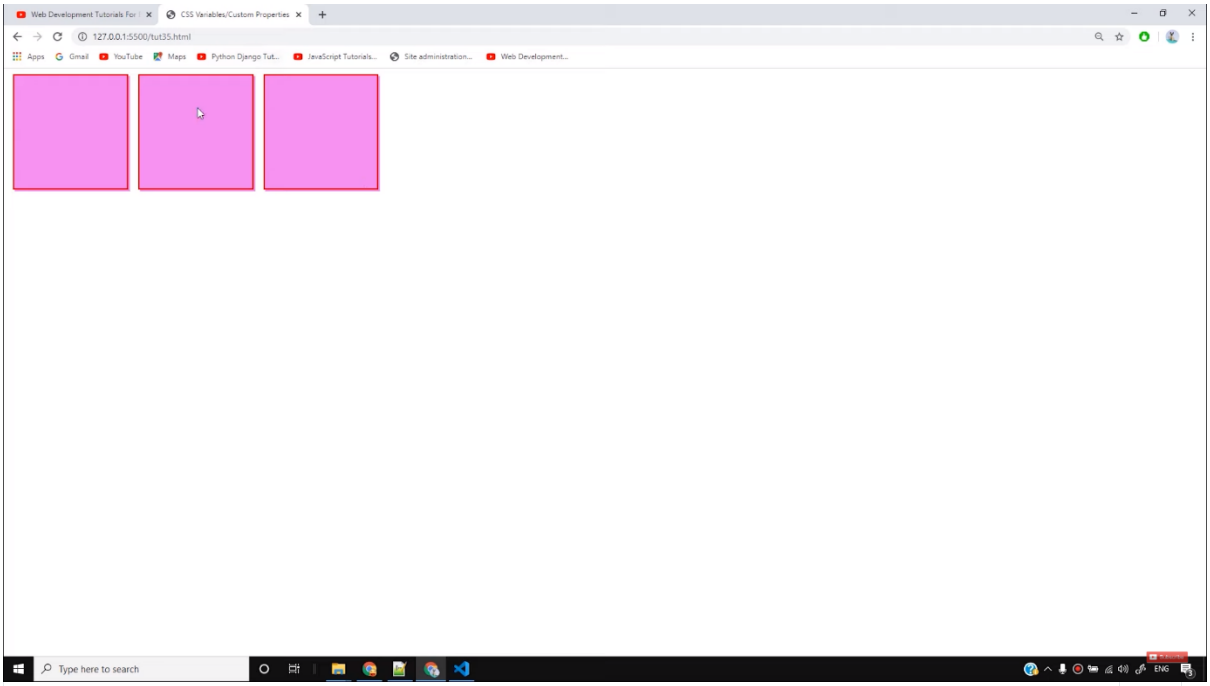


Now we will understand the concept of **variables.** Suppose, we want to create a variable for the background color. We can create it by **'--'** symbol. Variables in CSS helps us to assign the same properties to different elements. Let us analyze it with the code given below-

```
.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;
        }
```

Copy

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;
        }
```

Copy

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;
        }
```

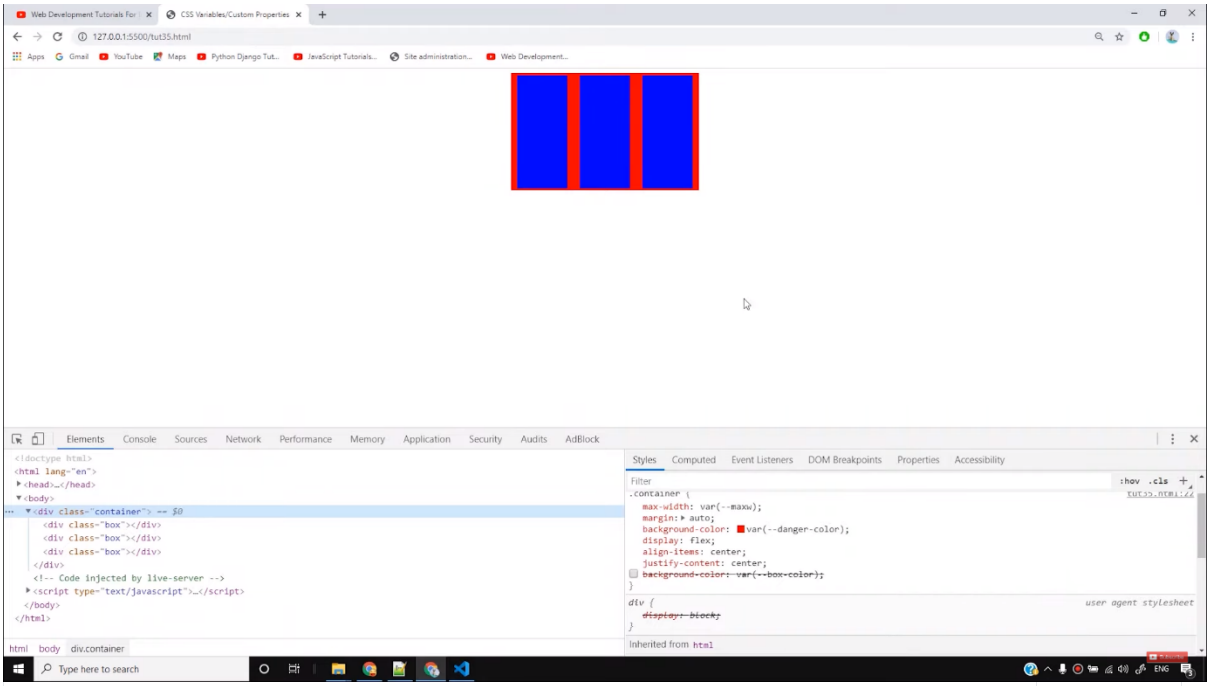
Copy

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); */
        }
```

Copy

The container will look as follows-



So, I believe you must have understood the concepts of variables and custom properties in CSS and how they can be used to minimize our efforts. For more interesting upcoming tutorials, kindly stay tuned and keep practicing.

Code as described/written in the video

```
<!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 Variables/Custom Properties</title>
    <style>
        :root{
            --primary-color: blue;
            --danger-color: red;
            --maxw: 333px;
        }
        .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;
        }
        .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); */
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box"></div>
        <div class="box"></div>
        <div class="box"></div>
    </div>
</body>
</html>
```

Keyframes and Animations in the <title> tag.
You will definitely appreciate the techniques shared in this tutorial about animations. Let us now write the HTML code for our website as follows-

```
<body>
```

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

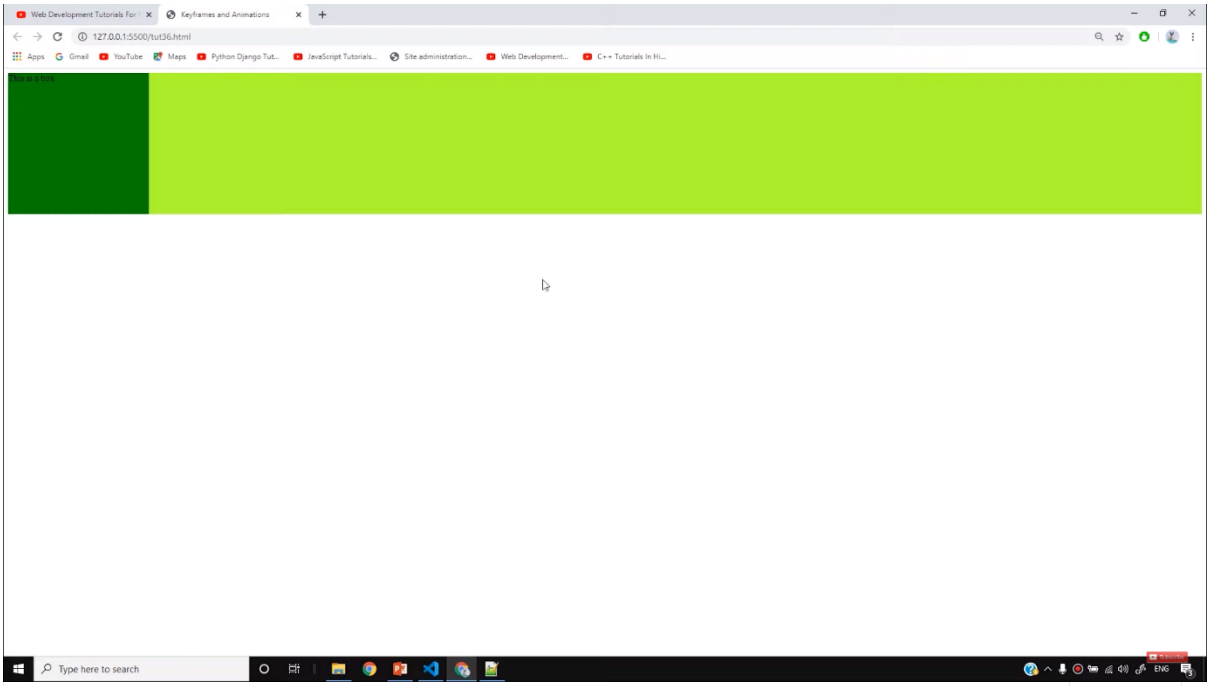
Copy

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;
}
```

Copy

The output will be as follows-



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: harry2;
        animation-duration: 8s;
        animation-iteration-count: 1;
}
```

Copy

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 harry1 {
        from {
            width: 200px;
        }

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

Copy

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;
```

Copy

- **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;
```

Copy

- **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;
```

Copy

There is another method of creating animation apart from keyframes. For this, we will give the name as *harry2*.

```
@keyframes harry2 {
        0%{
            top:0px;
            left:0px;
        }
        25%{
            top: 250px;
            left: 0px;
        }
        75%{
            top: 250px;
            left: 250px;
        }
        100%{
            top: 0px;
            left: 250px;
        }

    }
```

Copy

In this way, we can create the animations in terms of different percentages. We can assign different values here that will occur when that percentage of animation will be completed. Keep practicing these animations and in the upcoming tutorials, you will learn about practical implementations of these animations.

Code as described/written in the video

```
<!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>Keyframes and Animations</title>
</head>
<style>
    .container {
        background-color: greenyellow;
    }

    .box {
        background-color: green;
        width: 250px;
        height: 250px;
        position: relative;
        /* animation-name: harry1; */
        animation-name: harry2;
        animation-duration: 8s;
        animation-iteration-count: 1;
        /* animation-fill-mode: alternate; */
        /* animation-timing-function: ease-in-out; */
        /* animation-delay: 3s; */
        /* animation-direction: reverse;  */

        /* These properties can be set using this shorthand */
        /* animation: animation-name animation-duration animation-timing-function animation-delay animation-iteration-count animation-fill-mode; */
        /* animation: harry 5s ease-in 1s 12 backwards; */

    }
```

```
    @keyframes harry2 {
        0%{
            top:0px;
            left:0px;
        }
        25%{
            top: 250px;
            left: 0px;
        }
        75%{
            top: 250px;
            left: 250px;
        }
        100%{
            top: 0px;
            left: 250px;
        }

    }
    @keyframes harry1 {
        from {
            width: 200px;
        }

        to {
            width: 1400px;
        }
    }
</style>

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

```
        </div>
    </div>
</body>

</html>
```

CSS Transitions in the <title> tag. Let us now add some HTML code in the <body> tag to get started.

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

Copy

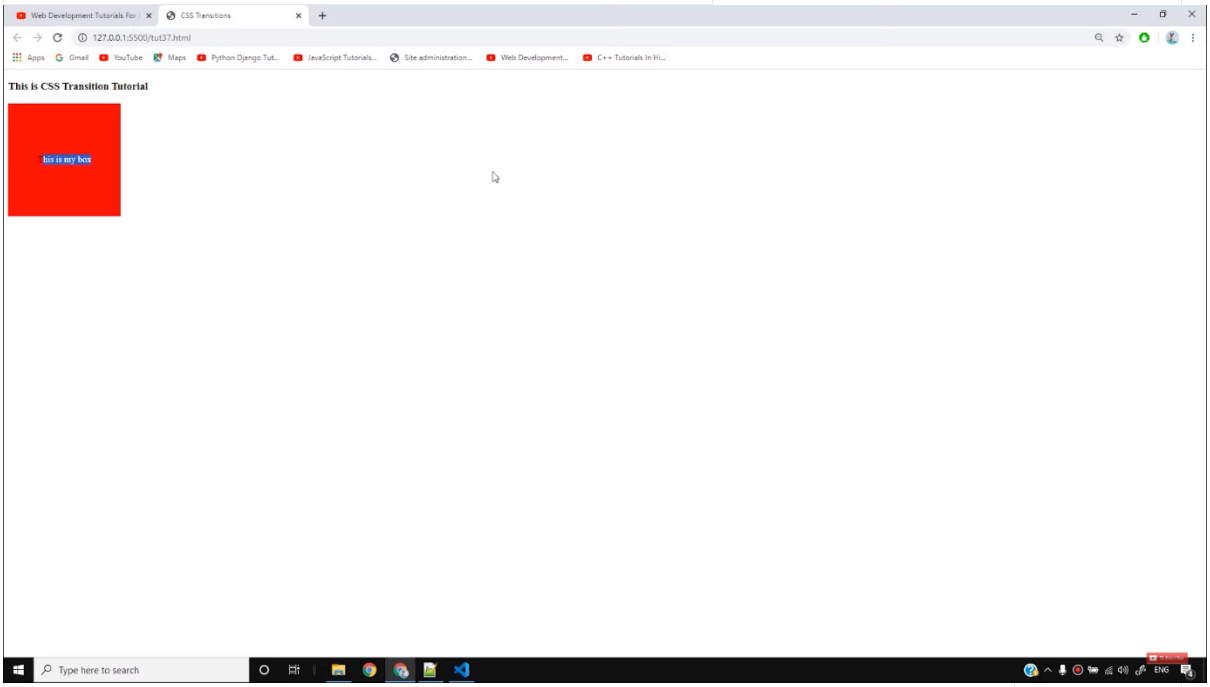
Now we will add some CSS in the box to see some of the transitions effects-

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

```
        align-items: center;
}
```

Copy

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



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;
}
```

Copy

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;
```

Copy

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;
```

Copy

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;
```

Copy

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;
```

Copy

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;
```

Copy

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

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

Copy

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;
    }
```

Copy

Here, all the properties will get changed accordingly and you witness some good transitions. All those properties that can change their values like colors, can show different transition

properties. You can try different such properties and view the effects of these transitions.

Code as described/written in the video

```
<!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 Transitions</title>
</head>
<style>
    body{
        background-color: black;
    }
    #box{
        display: flex;
        height: 200px;
        width: 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: 2s; */

        /* Transition short hand property in one line */
        /* transition: background-color 1s ease-in-out 2s; */


        transition: all 1s ease-in-out .3s;
```

```
    }

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

    }

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

Give the title as **CSS Transitions** in the <title> tag. Let us now add some HTML code in the <body> tag to get started.

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

```
        <div id="box">
            This is my box
        </div>
    </div>
</body>
```

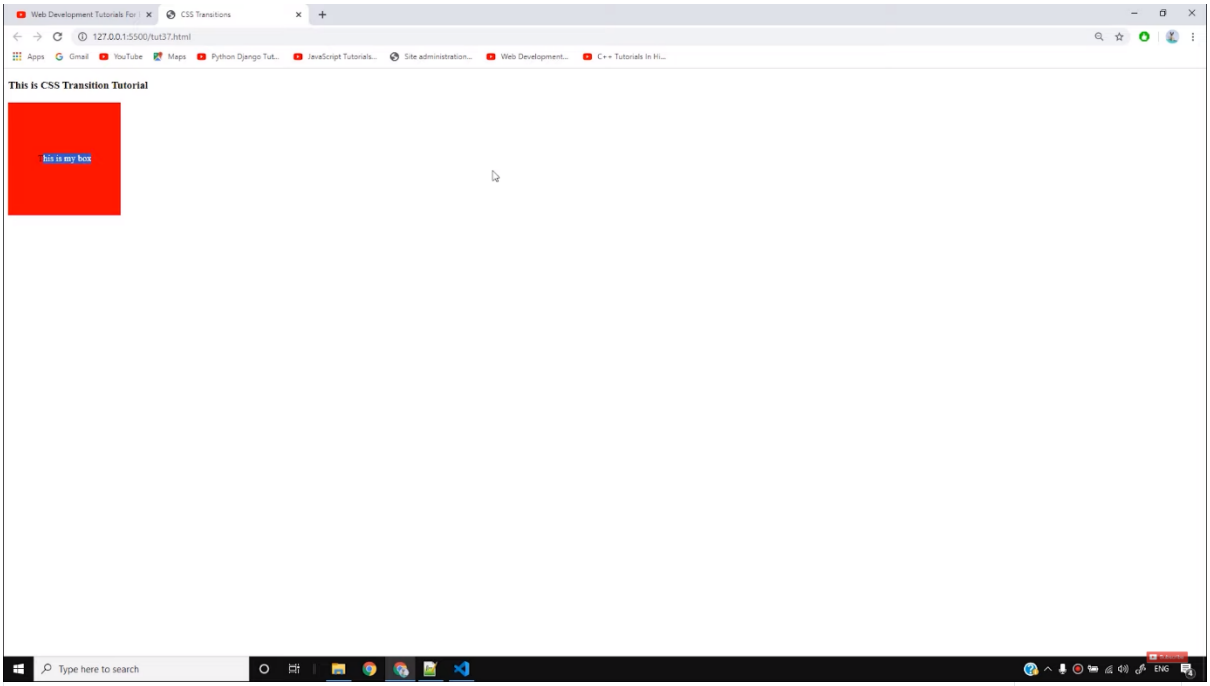
Copy

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;
}
```

Copy

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



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;
}
```

Copy

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;
```

Copy

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;
```

Copy

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;
```

Copy

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;
```

Copy

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;
```

Copy

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

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

Copy

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;
    }
```

Copy

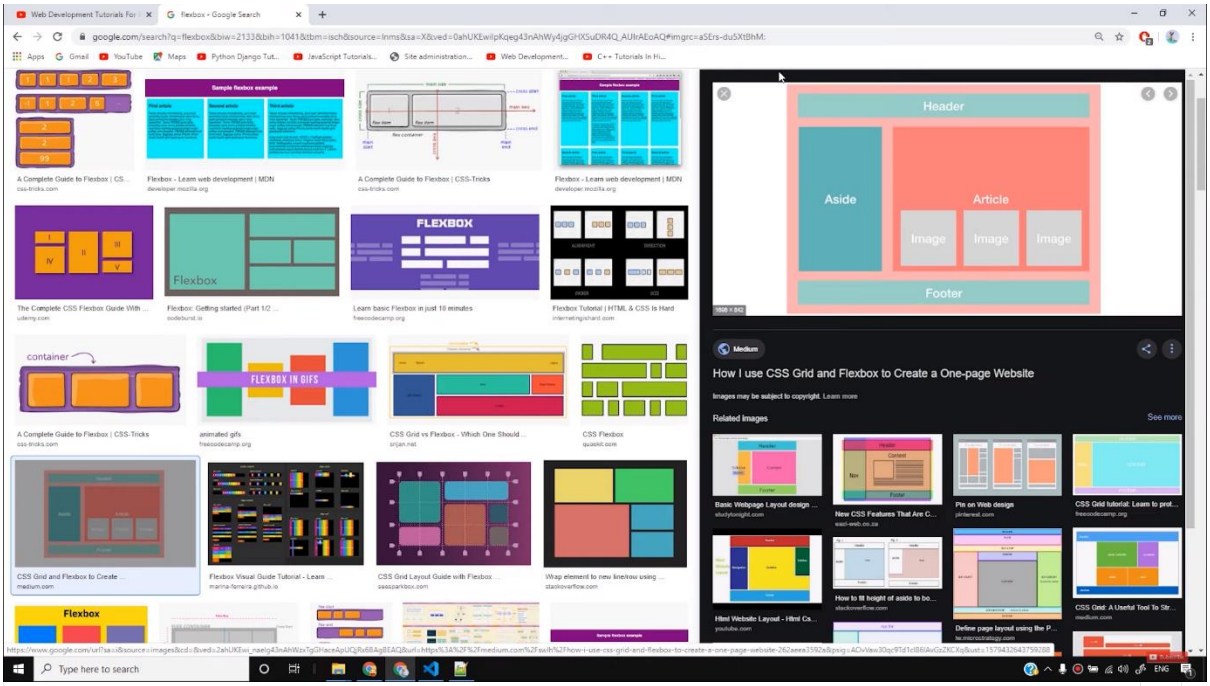
Here, all the properties will get changed accordingly and you witness some good transitions. All those properties that can change their values like colors, can show different transition properties. You can try different such properties and view the effects of these transitions.

Code as described/written in the video

```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 Transitions</title>
</head>
<style>
    body{
        background-color: black;
    }
    #box{
        display: flex;
        height: 200px;
        width: 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: 2s; */

        /* Transition short hand property in one line */
        /* transition: background-color 1s ease-in-out 2s; */


        transition: all 1s ease-in-out .3s;
    }

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

```
        height: 400px;
        width: 400px;
        border-radius: 100px;
        font-size: 45px;

    }

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

CSS Grid Tutorials under the <title> tag. 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 as follows-



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>
```

Copy

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;
}
```

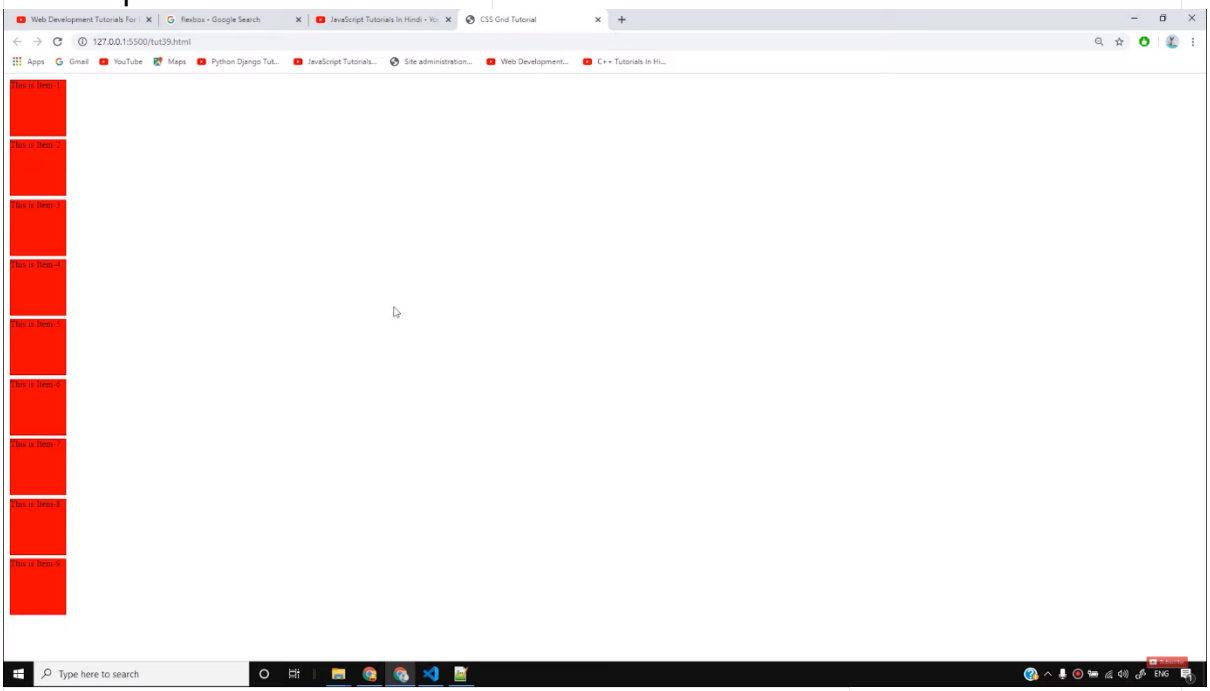
Copy

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-

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

Copy

The output will look as follows-

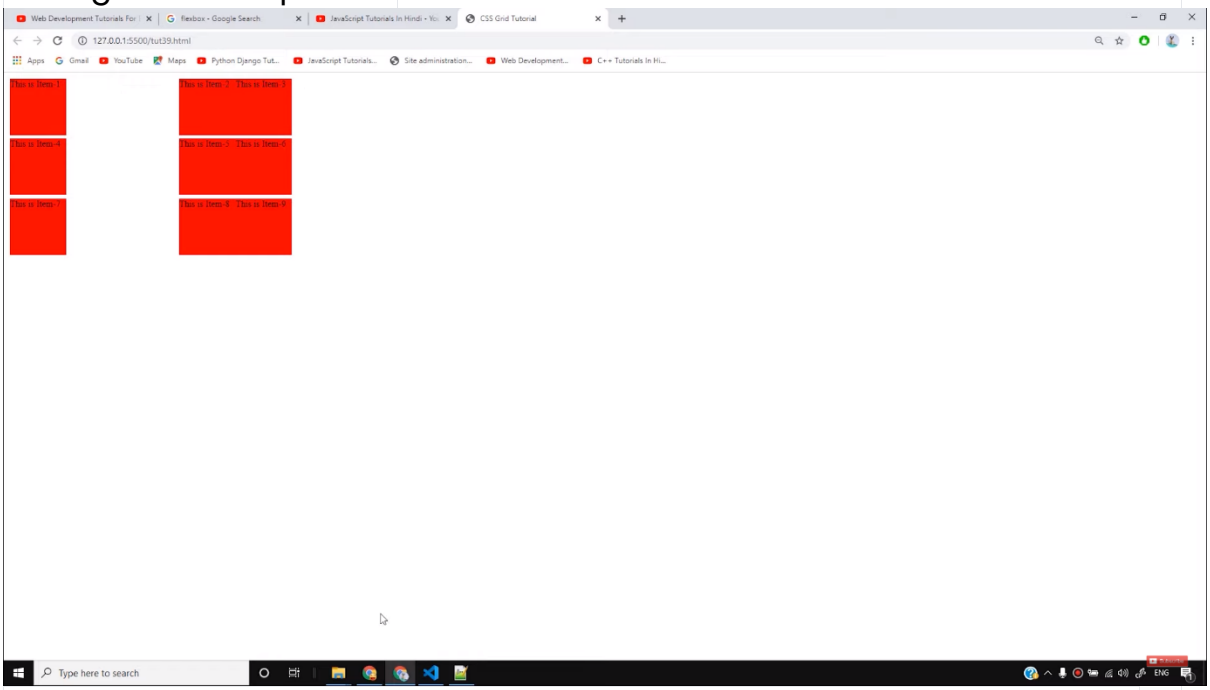


If we add the below code in the container as follows-

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

Copy

It will give the output as-



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;
```

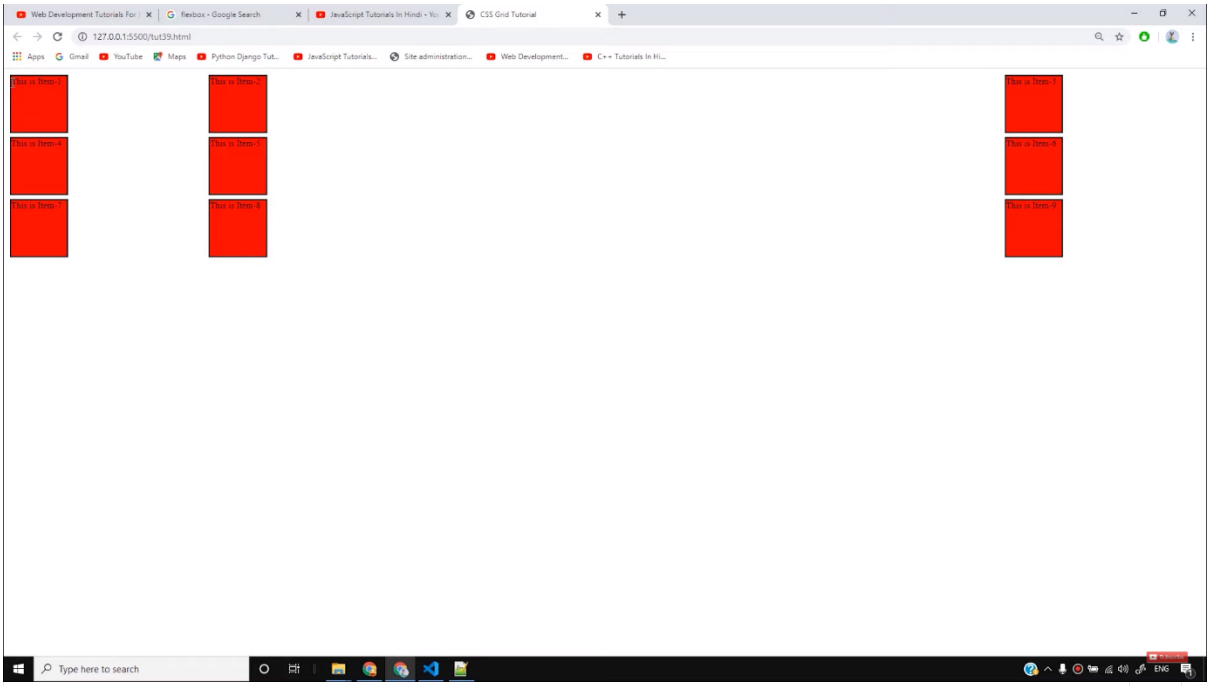
Copy

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

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

Copy

The above code means from the whole width, give 1 part to the first item, 4 parts to the second item, and again 1 part to the last item. The result will look as follows-



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

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

Copy

It will assign the properties depending upon the values provided.
In this tutorial, we have discussed how to make grids. From the next tutorials, we will learn the spanning of grids and how to merge them. Also, we will learn how to make them responsive using media queries. Till then stay with the tutorials.

Code as described/written in the video

```
<!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 Grid Tutorial</title>
    <style>
    .container{
```

```
        display: grid;
        /* grid-template-columns: 300px 100px 100px; */
        /* grid-template-columns: 300px auto 100px; */
        /* grid-template-columns: 1fr 4fr 1fr; */
        grid-template-columns: repeat(3, auto);
        grid-gap: 2rem;
    }
    .item{
        /* height: 100px;
        width: 100px; */
        background-color: red;
        border: 2px solid black;
        padding: 15px 5px;
    }
    </style>
</head>
<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 class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
```

```
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
        <div class="item">This is Item-9</div>
    </div>
</body>
</html>
```

give the title as **CSS Grid** under the <title> tag.
Lastly, we have discussed everything about grid columns and how to structure them but we have not seen anything about grid rows. Therefore, here we will learn to add different rows and set its length. Now let us write the HTML code. We will make a *class* with **grid** and add 10 *divs* inside it as follows-

```
<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>
```

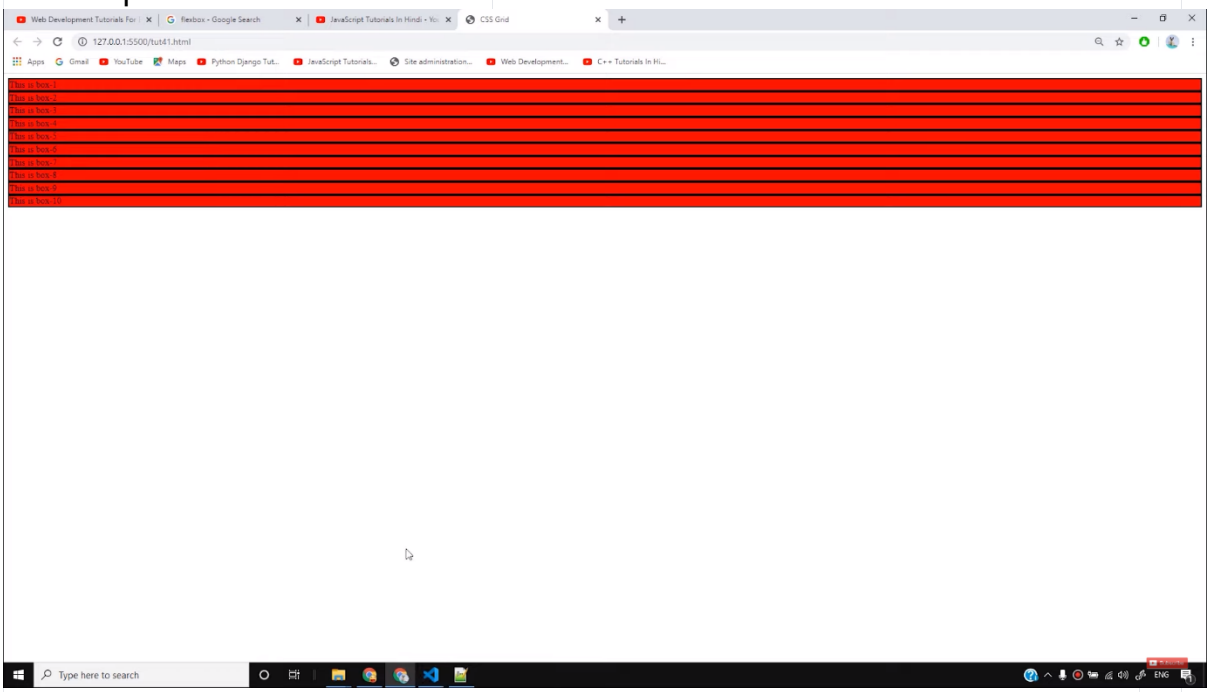
Copy

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

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

Copy

The output will look as follows-



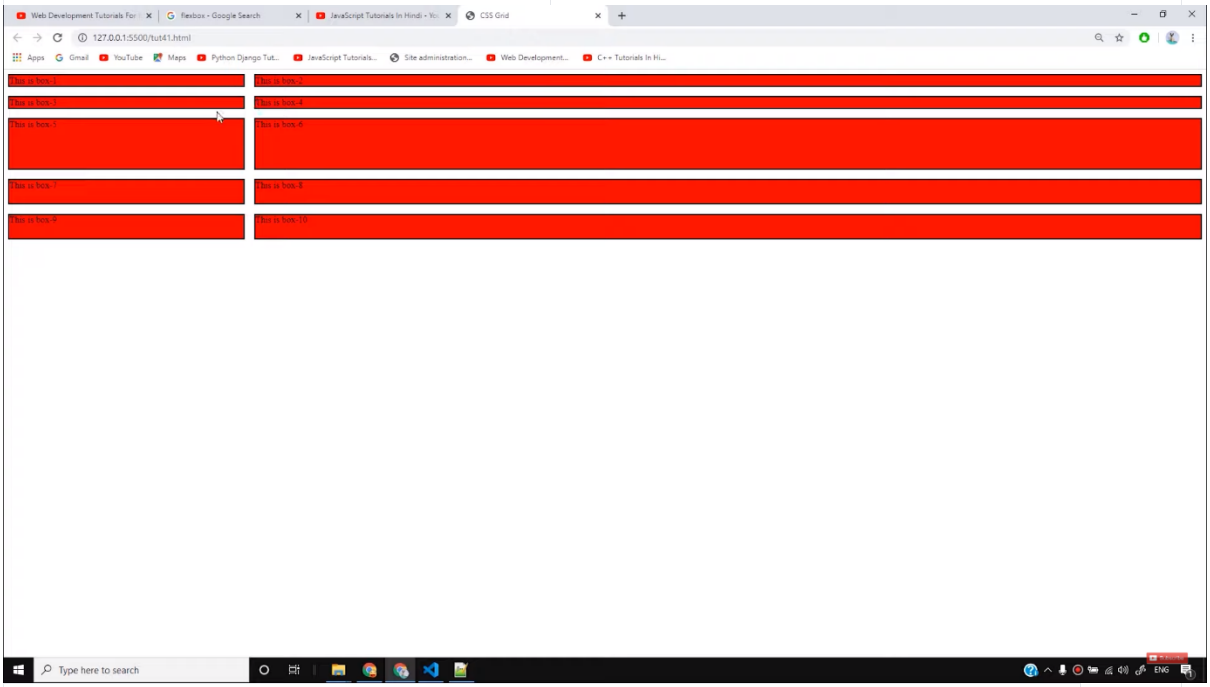
The next step is to display the grids and decide the number of rows in it and then customize it as follows-

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

Copy

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.
The result will now look as follows-



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-

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

Copy

This is how you can create grids of your own choice. With the help of the grid system, you can make your own grids, menus, or navigation bars without having any problem. From the next tutorial, we will learn the spanning of grids where we can assign different boxes to a particular

grid. Till then you can practice this to have good command over the grid system.

Code as described/written in the video

```
<!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 Grid </title>
    <style>
    .grid{
        display: grid;
        grid-template-rows: 1fr 1fr 4fr;
        grid-auto-rows: 2fr;
        grid-template-columns: repeat(4, 2fr);
        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>
```

Then give the title as **CSS Grid** under the <title> tag.
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>
```

Copy

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;
}
```

Copy

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;
}
```

Copy

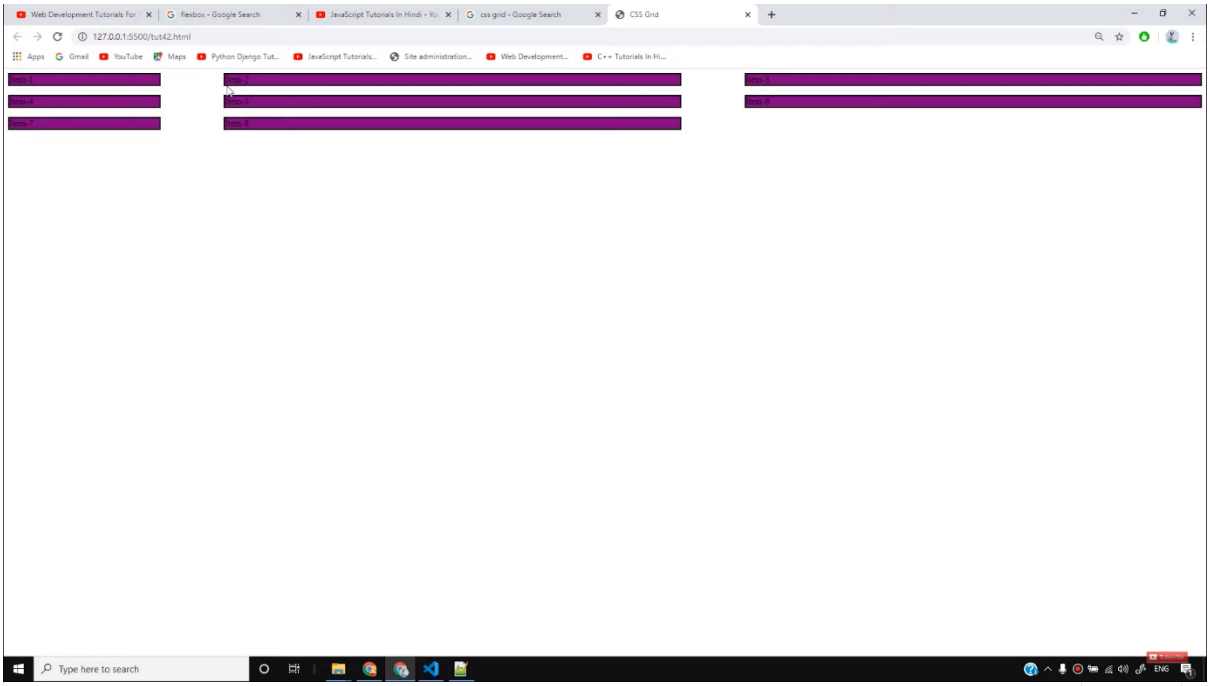
The output will look as follows-

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-

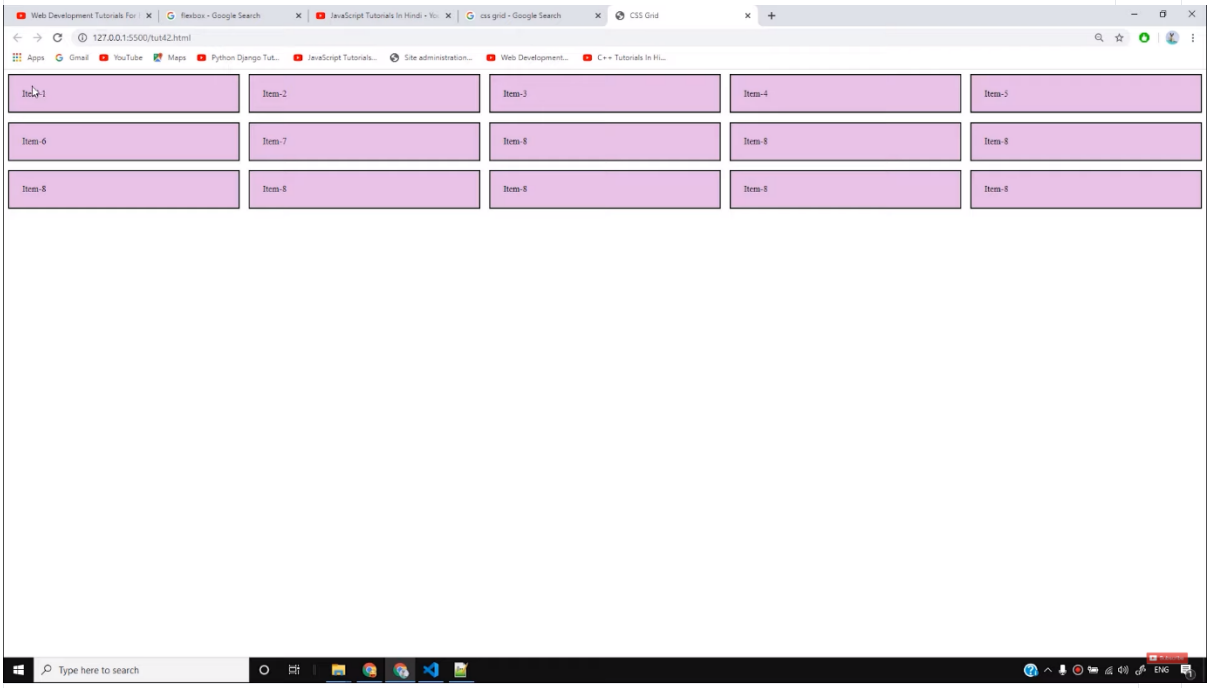
```
grid-column-gap: 7rem;
grid-row-gap: 1rem;
```

Copy

Now the output will look as follows-



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.



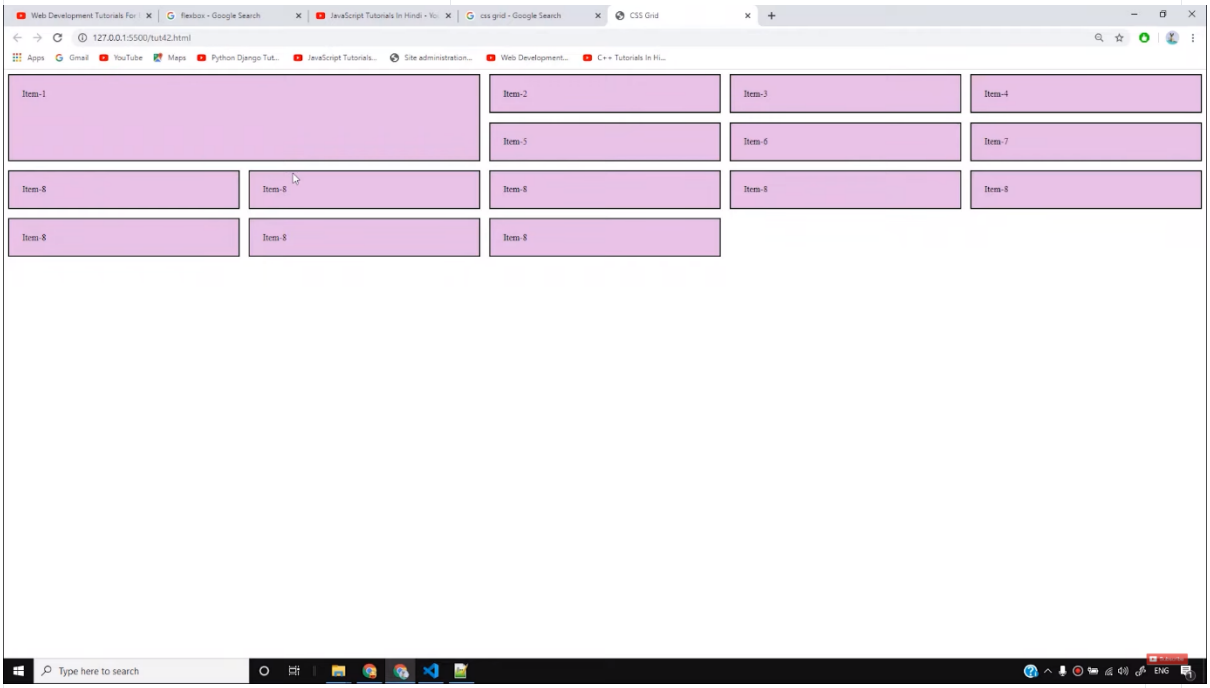
For doing this we can use the property called **grid row start and end.** If we write the code as follows-

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

```
        grid-row-end: 3;
}
```

Copy

The result will be as follows-



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;
    }
```

Copy

It will generate the same result as above.

One of the most confusion which now arises is what amongst *floats, flexbox* or *grids* should we use? The simple answer is, you are free to use any three of them irrespective of your website should look good and responsive and you should be comfortable with the development. You must always try to give the best user experience because it's the most important part of SEO.

Code as described/written in the video

```
<!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 Grid</title>
    <style>
    .container{
        display: grid;
        grid-template-columns: repeat(5, 1fr);
        grid-template-rows: repeat(4, 1fr);
        /* grid-column-gap: 7rem;
        grid-row-gap: 1rem; */
        grid-gap: 1rem;



    }
    .box{
        border: 2px solid black;
        background-color: rgb(228, 188, 228);
        padding: 23px;
    }
    .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;
    }
    </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-8</div>
        <div class="box">Item-8</div>
        <div class="box">Item-8</div>
        <div class="box">Item-8</div>
        <div class="box">Item-8</div>
        <div class="box">Item-8</div>
        <div class="box">Item-8</div>
    </div>
</body>
</html>
```

Then give the title as **CSS Grids** under the <title> tag. Through the HTML, we will add a container and 12 *divs* with the class **box** 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 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>
```

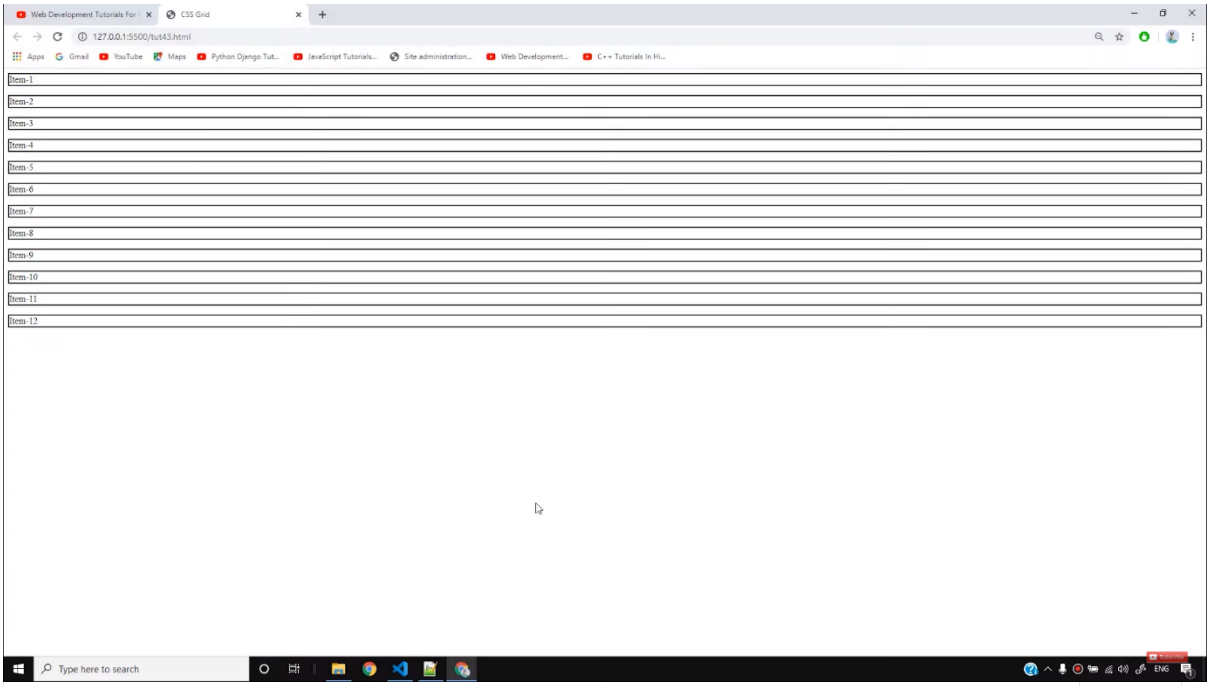
Copy

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

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

Copy

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

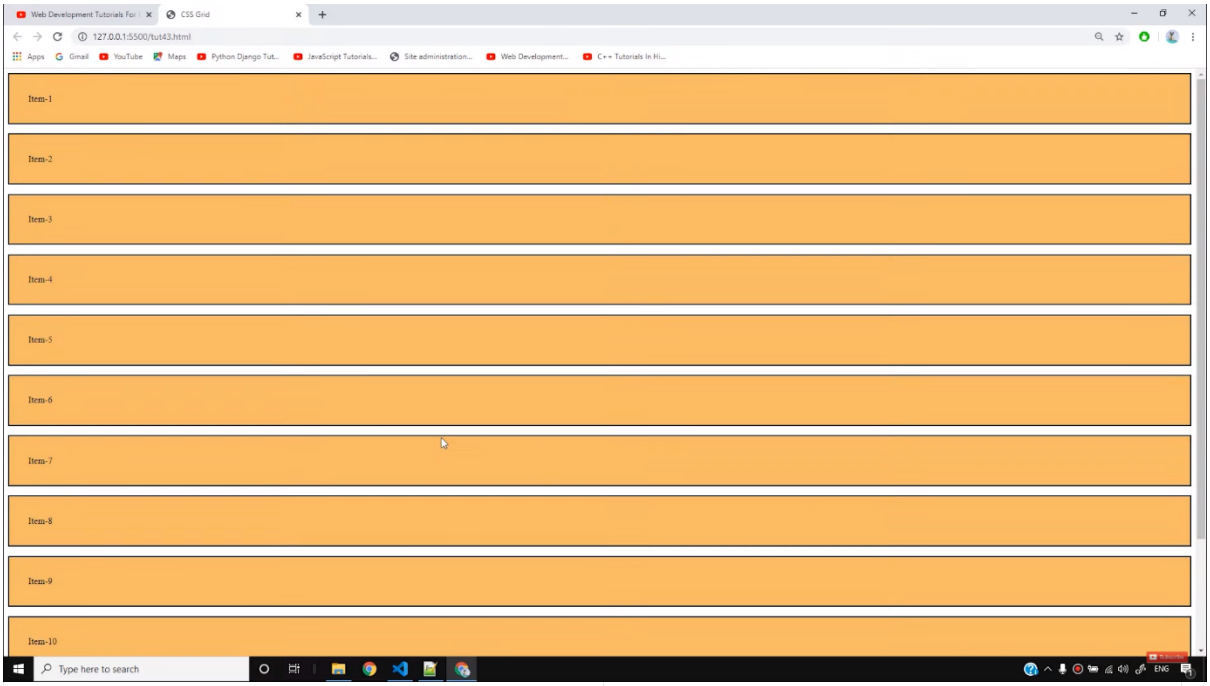


Now to give them the layouts of our choice, we can modify them by giving a background color and padding as follows-

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

Copy

The output will look as follows-

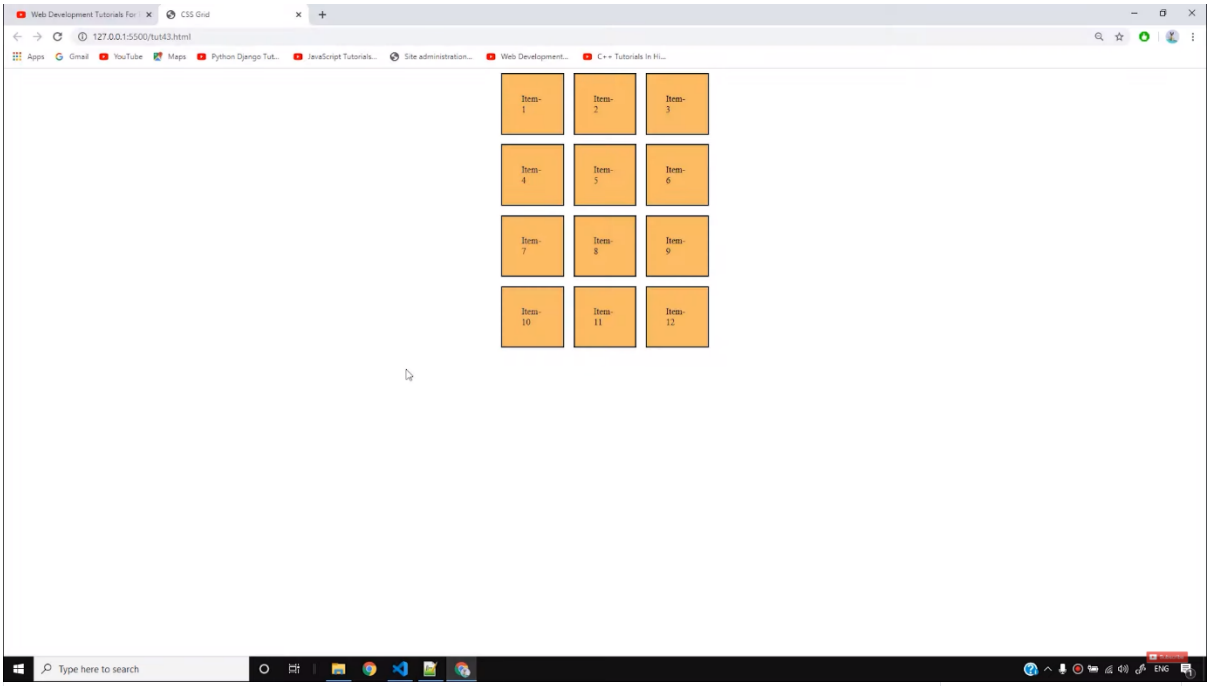


If we modify the container as follows-

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

Copy

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;
```

Copy

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;  */
        }
```

Copy

The benefit of **auto-fit** is that it will repeat the number of times it is necessary. The **minmax** will decide the minimum width given to any rows.

So I hope you have understood how to decide the minimum and maximum width of a row and make it responsive. You should keep practicing all the codes learned till now to make build your knowledge.

Code as described/written in the video

```
<!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 Grid</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: rgb(245, 187, 101);
            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 class="box">Item-13</div>
    </div>
</body>
</html>
```

give the title as **CSS Grid** under the <title> tag. Now let us understand what are grid template areas. To get started, we will add classes container and items as follows-

```
<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>
```

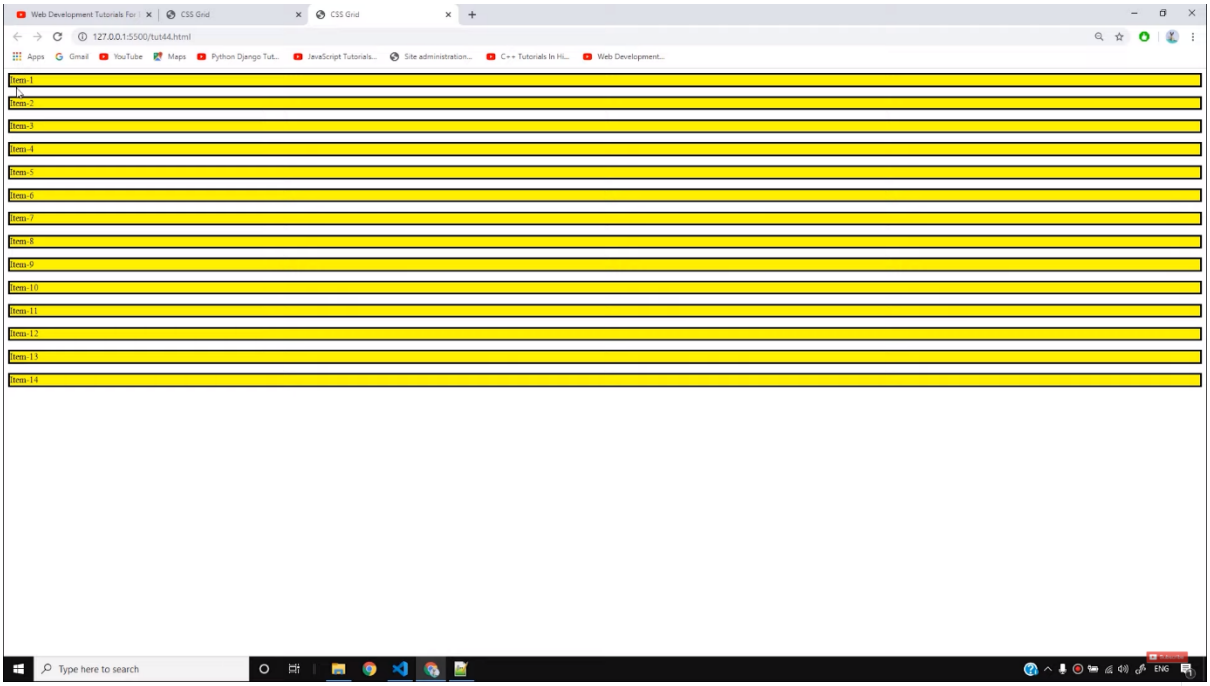
Copy

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;
}
```

Copy

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>
```

Copy

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;
```

```
    }
```

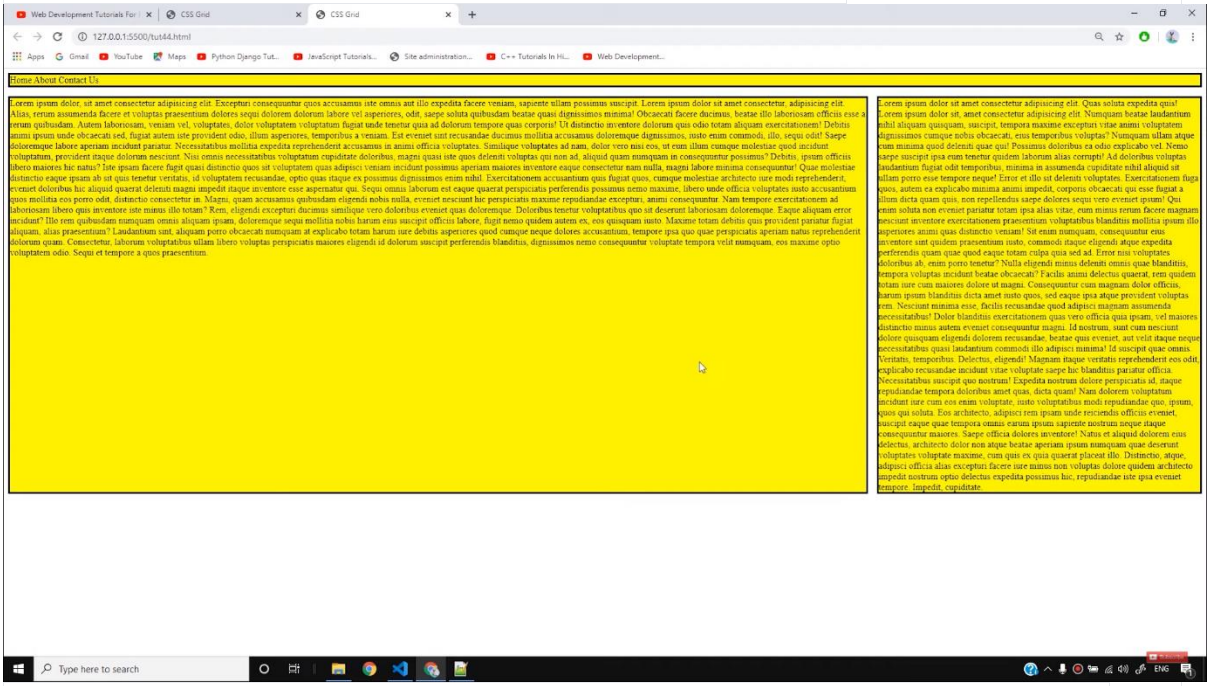
Copy

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'
}
```

Copy

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>
```

Copy

To add CSS, we can write-

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

Copy

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 ';
    }
```

Copy

CSS Grids is really a professional tool that minimizes our work. Therefore, you can command yourself over this by practicing and use it in making different layouts or sub layouts on the website.

Code as described/written in the video

```
<!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 Grid </title>
    <style>
    .container{
        display: grid;
        grid-gap: 1rem;
        grid-template-areas:
        'navbar navbar navbar navbar'
        'section section section aside'
        'footer footer footer footer ';
    }
    .item{
        background-color: yellow;
        border: 3px solid black;
        padding: 12px 23px;
    }
    #navbar{
        grid-area: navbar;
    }
    #section{
        grid-area: section;
    }
    #aside{
        grid-area: aside;
    }
   footer{
        grid-area: footer;
    }
    </style>
</head>
<body>
    <div class="container">
        <div id="navbar" class="item">

            Home About Contact Us
```

```
        </div>
        <div id="section" class="item">
            Lorem ipsum dolor, sit amet consectetur adipisicing elit. Excepturi consequuntur quos accusamus iste omnis aut illo expedita facere veniam, sapiente ullam possimus suscipit.
            Lorem ipsum dolor sit amet consectetur, adipisicing elit. Alias, rerum assumenda facere et voluptas praesentium dolores sequi dolorem dolorum labore vel asperiores, odit, saepe soluta quibusdam beatae quasi dignissimos minima! Obcaecati facere ducimus, beatae illo laboriosam officiis esse a rerum quibusdam. Autem laboriosam, veniam vel, voluptates, dolor voluptatem voluptatum fugiat unde tenetur quia ad dolorum tempore quas corporis! Ut distinctio inventore dolorum quis odio totam aliquam exercitationem! Debitis animi ipsum unde obcaecati sed, fugiat autem iste provident odio, illum asperiores, temporibus a veniam. Est eveniet sint recusandae ducimus mollitia accusamus doloremque dignissimos, iusto enim commodi, illo, sequi odit! Saepe doloremque labore aperiam incidunt pariatur. Necessitatibus mollitia expedita reprehenderit accusamus in animi officia voluptates. Similique voluptates ad nam, dolor vero nisi eos, ut eum illum cumque molestiae quod incidunt voluptatum, provident itaque dolorum nesciunt. Nisi omnis necessitatibus voluptatum cupiditate doloribus, magni quasi iste quos deleniti voluptas qui non ad, aliquid quam numquam in consequuntur possimus? Debitis, ipsum officiis libero maiores hic natus? Iste ipsam facere fugit quasi distinctio quos sit voluptatem quas adipisci veniam incidunt possimus aperiam maiores inventore eaque consectetur nam nulla, magni labore minima consequuntur! Quae molestiae distinctio eaque ipsam ab sit quis tenetur veritatis, id voluptatem recusandae, optio quas itaque ex possimus dignissimos enim nihil. Exercitationem accusantium quis fugiat quos, cumque molestiae architecto iure modi reprehenderit, eveniet doloribus hic aliquid quaerat deleniti magni impedit itaque inventore esse aspernatur qui. Sequi omnis laborum est eaque quaerat perspiciatis perferendis possimus nemo maxime, libero unde officia voluptates iusto accusantium quos mollitia eos porro odit, distinctio consectetur in. Magni, quam accusamus quibusdam eligendi nobis nulla, eveniet nesciunt hic perspiciatis maxime repudiandae excepturi, animi consequuntur. Nam tempore exercitationem ad laboriosam libero quis inventore iste minus illo totam? Rem, eligendi excepturi ducimus similique vero doloribus eveniet quas doloremque. Doloribus tenetur voluptatibus quo sit deserunt laboriosam doloremque. Eaque aliquam error incidunt? Illo rem quibusdam numquam omnis aliquam ipsam, doloremque sequi mollitia nobis harum eius suscipit officiis labore, fugit nemo quidem autem ex, eos quisquam iusto. Maxime totam debitis quis provident pariatur fugiat aliquam, alias praesentium? Laudantium sint, aliquam porro
```

obcaecati numquam at explicabo totam harum iure debitis asperiores quod cumque neque dolores accusantium, tempore ipsa quo quae perspiciatis aperiam natus reprehenderit dolorum quam. Consectetur, laborum voluptatibus ullam libero voluptas perspiciatis maiores eligendi id dolorum suscipit perferendis blanditiis, dignissimos nemo consequuntur voluptate tempora velit numquam, eos maxime optio voluptatem odio. Sequi et tempore a quos praesentium.

```
        </div>

        <div id="aside" class="item">Lorem ipsum dolor sit amet consectetur adipisicing elit. Quas soluta expedita quis!

            Lorem ipsum dolor sit, amet consectetur adipisicing elit. Numquam beatae laudantium nihil aliquam quisquam, suscipit, tempora maxime excepturi vitae animi voluptatem dignissimos cumque nobis obcaecati, eius temporibus voluptas? Numquam ullam atque cum minima quod deleniti quae qui! Possimus doloribus ea odio explicabo vel. Nemo saepe suscipit ipsa eum tenetur quidem laborum alias corrupti! Ad doloribus voluptas laudantium fugiat odit temporibus, minima in assumenda cupiditate nihil aliquid sit ullam porro esse tempore neque! Error et illo sit deleniti voluptates. Exercitationem fuga quos, autem ea explicabo minima animi impedit, corporis obcaecati qui esse fugiat a illum dicta quam quis, non repellendus saepe dolores sequi vero eveniet ipsum! Qui enim soluta non eveniet pariatur totam ipsa alias vitae, eum minus rerum facere magnam nesciunt inventore exercitationem praesentium voluptatibus blanditiis mollitia ipsum illo asperiores animi quas distinctio veniam! Sit enim numquam, consequuntur eius inventore sint quidem praesentium iusto, commodi itaque eligendi atque expedita perferendis quam quae quod eaque totam culpa quia sed ad. Error nisi voluptates doloribus ab, enim porro tenetur? Nulla eligendi minus deleniti omnis quae blanditiis, tempora voluptas incidunt beatae obcaecati? Facilis animi delectus quaerat, rem quidem totam iure cum maiores dolore ut magni. Consequuntur cum magnam dolor officiis, harum ipsum blanditiis dicta amet iusto quos, sed eaque ipsa atque provident voluptas rem. Nesciunt minima esse, facilis recusandae quod adipisci magnam assumenda necessitatibus! Dolor blanditiis exercitationem quas vero officia quia ipsam, vel maiores distinctio minus autem eveniet consequuntur magni. Id nostrum, sunt cum nesciunt dolore quisquam eligendi dolorem recusandae, beatae quis eveniet, aut velit itaque neque necessitatibus quasi laudantium commodi illo adipisci minima! Id suscipit quae omnis. Veritatis, temporibus. Delectus, eligendi! Magnam itaque veritatis reprehenderit eos odit, explicabo recusandae incidunt vitae voluptate saepe hic blanditiis pariatur officia. Necessitatibus suscipit quo nostrum! Expedita nostrum dolore perspiciatis id, itaque repudiandae tempora doloribus amet quas, dicta quam! Nam dolorem voluptatum incidunt iure cum eos enim voluptate, iusto
```

```
voluptatibus modi repudiandae quo, ipsum, quos qui soluta. Eos
architecto, adipisci rem ipsam unde reiciendis officiis eveniet,
suscipit eaque quae tempora omnis earum ipsum sapiente nostrum neque
itaque consequuntur maiores. Saepe officia dolores inventore! Natus
et aliquid dolorem eius delectus, architecto dolor non atque beatae
aperiam ipsum numquam quae deserunt voluptates voluptate maxime, cum
quis ex quia quaerat placeat illo. Distinctio, atque, adipisci
officia alias excepturi facere iure minus non voluptas dolore quidem
architecto impedit nostrum optio delectus expedita possimus hic,
repudiandae iste ipsa eveniet tempore. Impedit, cupiditate.
        </div>
        <footer class="item">Lorem, ipsum dolor sit amet consectetur
adipisicing elit. Provident quo libero cumque.</footer>
        <!-- <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>
</html>
```

Give the title as **CSS Grid + Media Queries** under the <title> section.

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;
        }
    }
```

Copy

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>
```

Copy

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

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

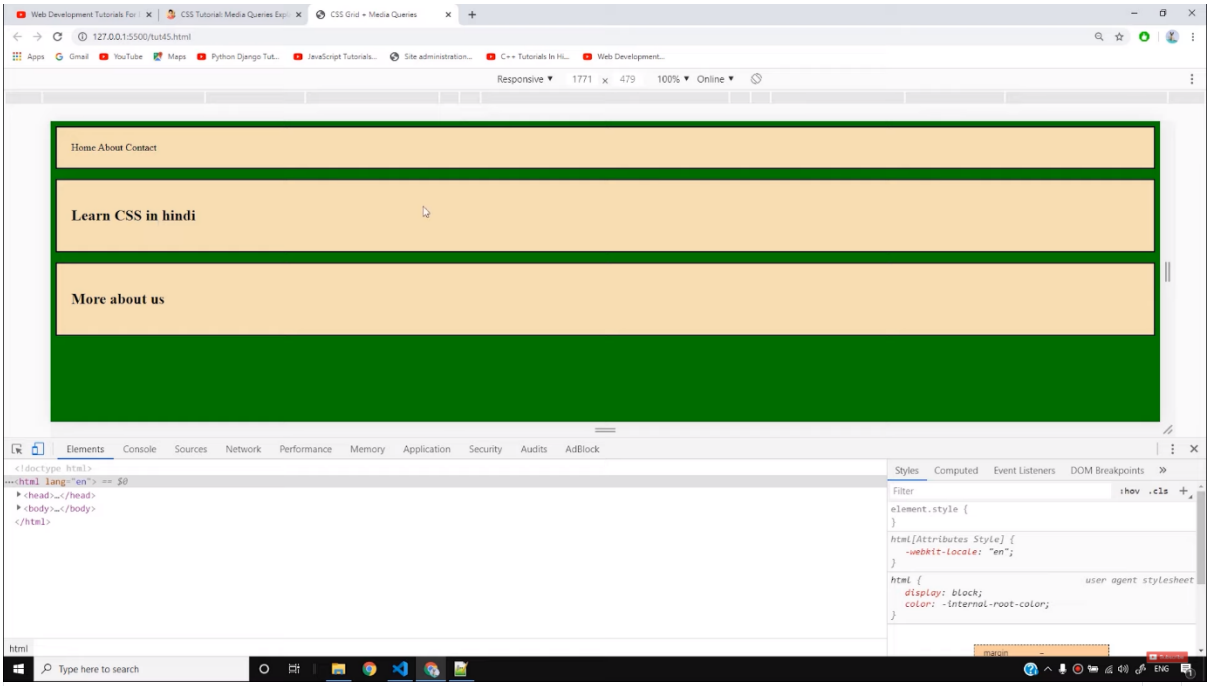
Copy

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;
    }
```

Copy

The result will look as follows-



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 ';
```

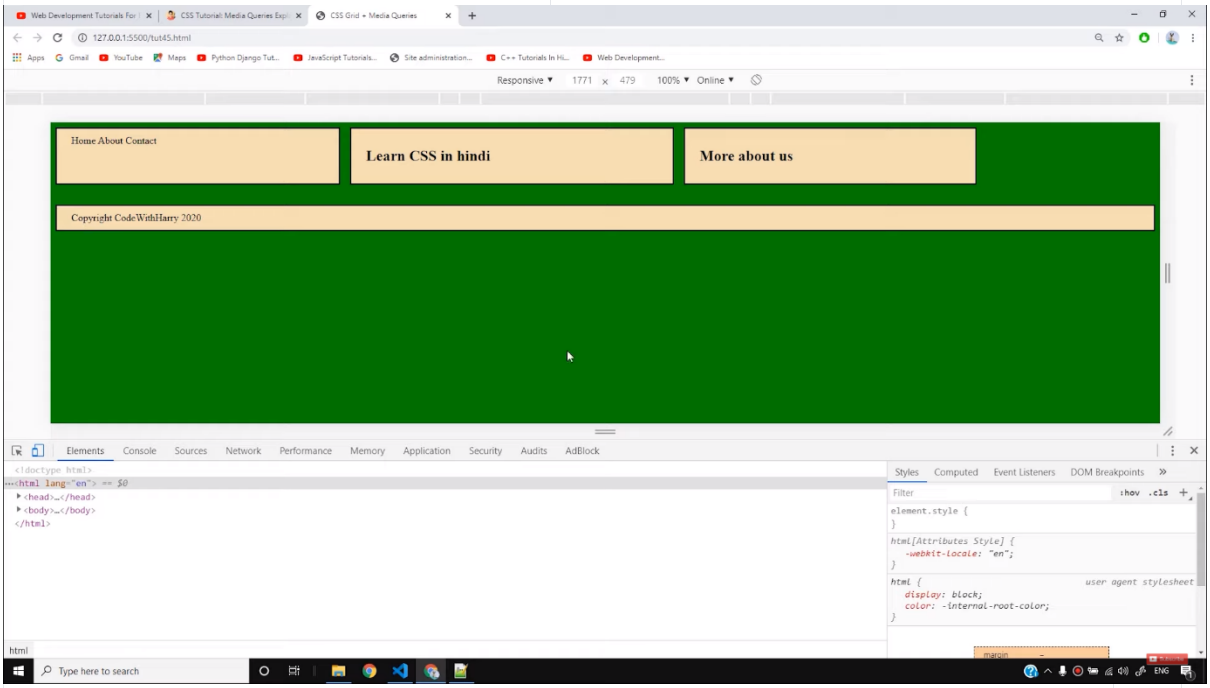
Copy

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

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

Copy

The result will now look as follows-



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 ';
        }
```

Copy

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 ';
        }
```

Copy

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;
        }
```

Copy

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 as described/written in the video

```
<!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 Grid + Media Queries</title>
</head>
<style>
    .container {
        display: grid;
        grid-gap: 1rem;
        grid-template-areas:
            'navbar navbar navbar navbar'
```

```
            'section section section aside'
            'footer footer footer footer ';
    }

    @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'
                'footer footer footer footer ';
        }
        aside{
            display: none;
        }
        span{
            display: block;
            text-align: center;
        }
    }

    @media only screen and (min-width: 300px) and (max-width:500px)
{
        body {
            background-color: blue;
        }
        .container {
            grid-template-areas:
                'navbar navbar navbar navbar'
                'section section section section'
                'aside aside aside aside'
```

```
                'footer footer footer footer ';
        }
        aside{
            display: none;
        }
        span{
            display: block;
            text-align: center;
        }
    }

    @media (min-width: 500px) and (max-width:800px) {
        body {
            background-color: yellow;
        }

        .container {
            grid-template-areas:
                'navbar navbar navbar navbar'
                'section section section section'
                'aside aside aside aside'
                'footer footer footer footer ';
        }
        span{
            display: block;
            text-align: center;
        }
    }

    @media (min-width: 800px) {
        body {
            background-color: green;
        }
    }
```

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

    nav {
        grid-area: navbar;
    }

    section {
        grid-area: section;
    }

    aside {
        grid-area: aside;
    }

    footer {
        grid-area: footer;
        text-align: center;
    }
</style>

<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>
            <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Doloremque nemo cupiditate distinctio et expedita. Delectus quaerat accusamus inventore tenetur error quam minus, provident fugit repellat necessitatibus? Architecto itaque quidem sequi blanditiis, facere accusantium hic.lorem122 Lorem ipsum dolor sit amet consectetur, adipisicing elit. Eligendi dignissimos ut rerum aut. Fuga delectus suscipit debitis libero. Itaque, ipsum dignissimos consequatur repellat distinctio enim dolorem, facilis esse dolorum illum earum aliquid numquam blanditiis ipsam reiciendis iure nobis quo cum maiores aperiam pariatur. Quas aliquam, quae non rerum, architecto eligendi blanditiis officia placeat dolor soluta explicabo. Voluptatem tenetur perspiciatis neque quidem ducimus velit id explicabo, illo magni quis voluptatum. Tempora quod, dicta illum ratione quo at dolores cupiditate aperiam laboriosam amet sequi quaerat similique incidunt eius deleniti deserunt accusamus eligendi nemo est veritatis fugiat ducimus! Modi ut vel et nihil asperiores mollitia obcaecati, neque accusantium corrupti, quisquam voluptatem rem? Possimus tempore et fugit cumque culpa aliquam doloremque odio hic, cum, minima nostrum!</p>
        </section>
        <aside class="bdr">
            <h1>More about us</h1>
        </aside>
    </div>
    <footer class="bdr">Copyright CodeWithHarry 2020</footer>

</body>

</html>
```