HTML and CSS

**Attributes**

Attributes are used to give extra information about the element. It comes in name and value pairs.

e.g

<img> is a tag

<img src=””> src is an attribute.

**Relative path**

/ 🡪 root directory

../ 🡪 parent directory (one level up the folder)

./ 🡪 current directory

Images/a.png 🡪 pointing to images folder in current directory.

Viewport: Area of the screen where elements can be seen.

**Emphasis vs Italics**

<em> 🡪 It is used to emphasize important words or section of words. The screen reader will pronounce the emphasize text will a verbal stress.

<i> 🡪 It is just for styling purpose.

**<colgroup>**

colgroup are used to group one or more coloumns in table for formatting.

Colgroup must be before thead, tbody and after table, caption tag.

Styling applied in sequential order.

**<col>**

Within colgroup we use col to access the coloumns.

span 🡪 specifies number of coloumns to span.

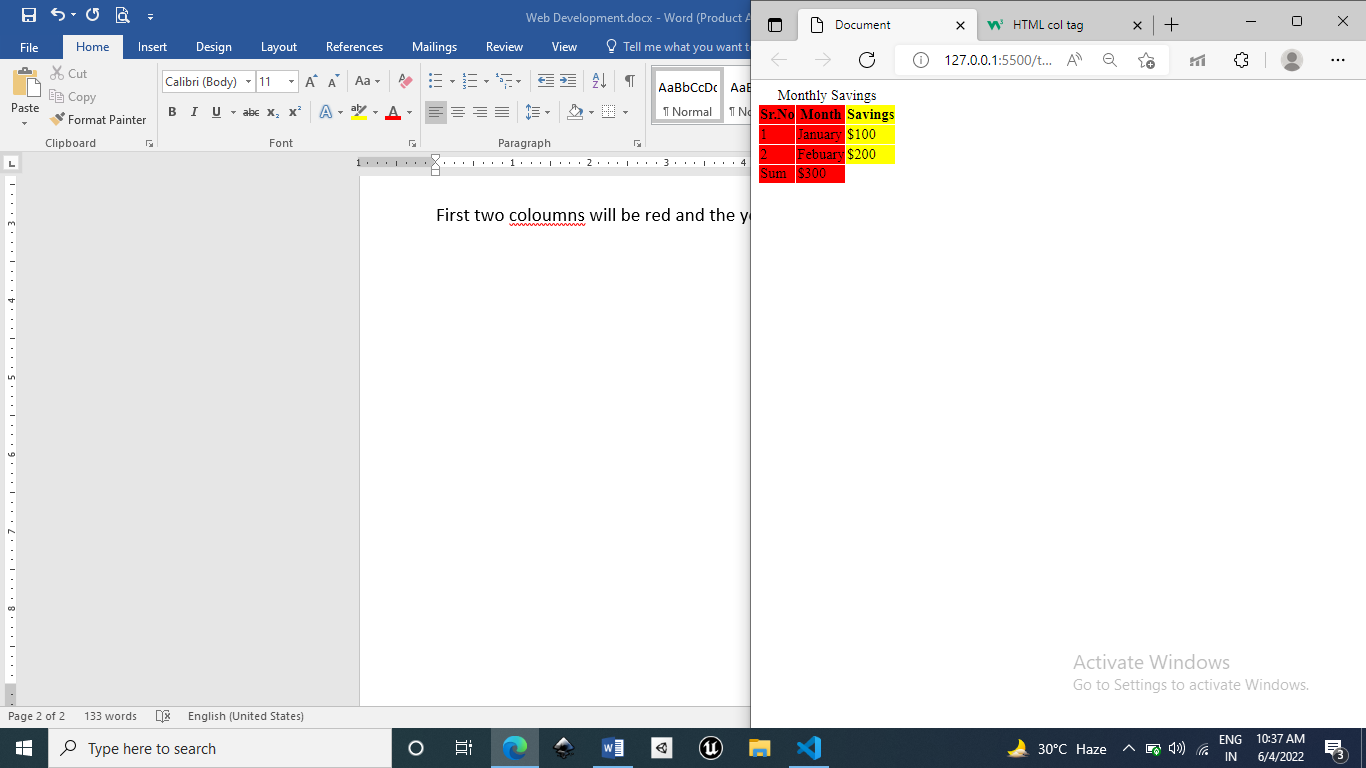
<colgroup>

    <col span="2" "background-color: red;">

    <col style="background-color: yellow;">

</colgroup>

First two coloumns will be red and the yellow.



Forms

<form>

<label for=”id\_of\_input\_element”>First name</label>

<input type=”text” id=”id\_of\_input\_element”>

</form>

Formatting Elements

* <b> - Bold text but not important
* <strong> - Important text and bold
* <i> - Italic text (just styling)
* <em> - Emphasized text (gives verbal stress)
* <mark> - Marked text / Highlighting the text
* <small> - Smaller text
* <del> - Deleted text / Text with a cut(~~deleted~~)
* <ins> - Inserted text/ Text with a underline(insert)
* <sub> - Subscript text
* <sup> - Superscript text

<picture> 🡪 display different elements according to screen sizes. This element contains one or more source elements.

<source> 🡪 this has attributes

“media” 🡪 has condition when a particular image must be shown.

“srcset” 🡪 has image url

**Descripiton List**

<dl> 🡪 define description list

<dt> 🡪 define term in list

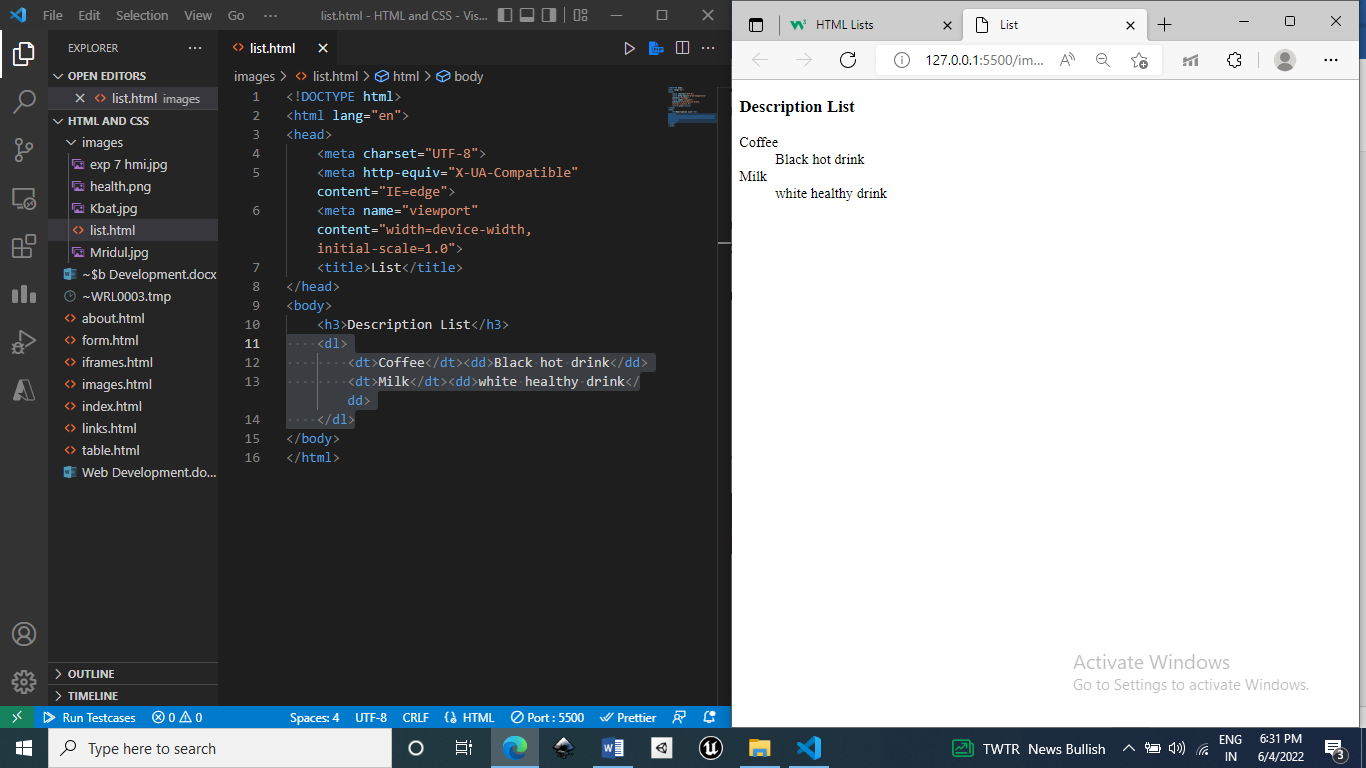
<dd> 🡪 define description of the term in list.

    <dl>

        <dt>Coffee</dt><dd>Black hot drink</dd>

        <dt>Milk</dt><dd>white healthy drink</dd>

    </dl>



**Block vs Inline elements**

**Block** 🡪 Starts at new line. Takes up as much width as much possible(stretches left and right). <p>, <div>

Width and height, margin padding properties works with block.

**Inline** 🡪 Doesn’t starts at new line and takes as much space as needed. <span><a><i><b><bdo>

Width, height properties doesn’t works with inline.

Margin top and bottom doesn’t works with inline.

**Inline-Block Elements**

We can edit the height, width, margin top and margin bottom property of the element and element does not start from the new line and takes as much width as required by default.

**Viewport**

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

**Can We Omit <html> and <body> tag???**

Yes, we can omit both the tags, HTML page will validate the page.

But, omitting <body> will create problems in older browser.

Omitting <html> and <body> tags will crash DOM and XML

Omitting <head>?

The HTML <head> tag can also be omitted.

Browsers will add all elements before <body>, to a default <head> element.

**HTML Entities**

Character entities are used to display reserved characters in HTML.

“<” or ”>” is a reserved character so we use “&lt” to print “<”

“&nsbp” 🡪 non break space

**<base>** 🡪 defines base url for all relative URLs.

<base href="https://www.w3schools.com/" target="\_blank">

<a href="tags/tag\_base.asp">HTML base Tag</a>

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<code>](https://www.w3schools.com/tags/tag_code.asp) | Defines programming code |
| [<kbd>](https://www.w3schools.com/tags/tag_kbd.asp) | Defines keyboard input |
| [<samp>](https://www.w3schools.com/tags/tag_samp.asp) | Defines computer output |
| [<var>](https://www.w3schools.com/tags/tag_var.asp) | Defines a variable |

**CSS Property**

**White space 🡪 how to deal with white space of the element.**

white-space : nowrap (means that the words/content in the element wont be be wrapped means all the conent/words will be on the single line and we would need horizontal bar to see all the content. White space will be not preserved. **Sequence of whitespaces will be collapsed to one whitespace.**)

white-space: pre (means the content will be displayed as it is written in html body. Means white space will be preserved. **Sequence of whitespace will not be collapsed to one whitespace.**)

white-space: normal (**Sequence of whitespaces will be collapsed to one whitespace.**) the content/words will be wraped.

Position:

**Static(default)**: would not get affected top, bottom, left, right.

Positioned acc to normal flow of the image.

**Relative**: Element is positioned relative to its normal position.

**Absolute**: element is position relative to its parent position only if the parent position is relative or else it will be relative to body by default.

**Fixed**: Positioned relative to viewport(area which is visible to user).

**Sticky:** Toggles between relative and fixed. It is positioned acc to user’s scroll position.

It is first relative to the normal position but when user scrolls and if the given offset position is met in viewport area then its position attribute becomes “fixed”.

look for any **overflow** property set on the parent. **You can't use**: **overflow: hidden**, **overflow: scroll** or **overflow: auto** on the parent of a position: sticky element.

Z-index

Specifies which element should appear at front or back.

Element 1 z-index: 1(front) (front element 2 & 3)

Element 2 z-index: 0(mid) (behind element 1 & front of element 3)

Element 3 z-index: -1(back) (behind element 1 & 2)

Overflow

(This property decides whether to cut the content or add scrollbars to the element if content is too big to fit in specified area.)

Auto: scrollbar will appear if required.

Scroll: horizontal and vertical scroll bar will required.

Hidden: content which will go out of the box will not be visible.

Visible: whole content will be visible even if it goes outside the box.

Float

Use to specify how element must be positioned and formatted.

Clear

Elements next to a floating element will flow around it.

It specifies weather the element next to the float elements to be below floating element so we use clear property.

When we use the float property, and we want the next element below (not on right or left), we will have to use the clear property.

The clear property specifies what should happen with the element that is next to a floating element.

The clear property can have one of the following values:

* none - The element is not pushed below left or right floated elements. This is default
* left - The element is pushed below left floated elements
* right - The element is pushed below right floated elements
* both - The element is pushed below both left and right floated elements
* inherit - The element inherits the clear value from its parent

Pseudo Classes

It specifies about the special state of the element. E.g for links hover, visited, active, link. E.g first-child, last-child. E.g for input, focus.

a:link{}

a:visited{}

a:hover{}

a:active{}

input:focus{}

p:first-child{}

p:last-child{}

Syntax:

selector:class

{ property: value;

}

Pseudo-Element

**Style particular part of the element.**

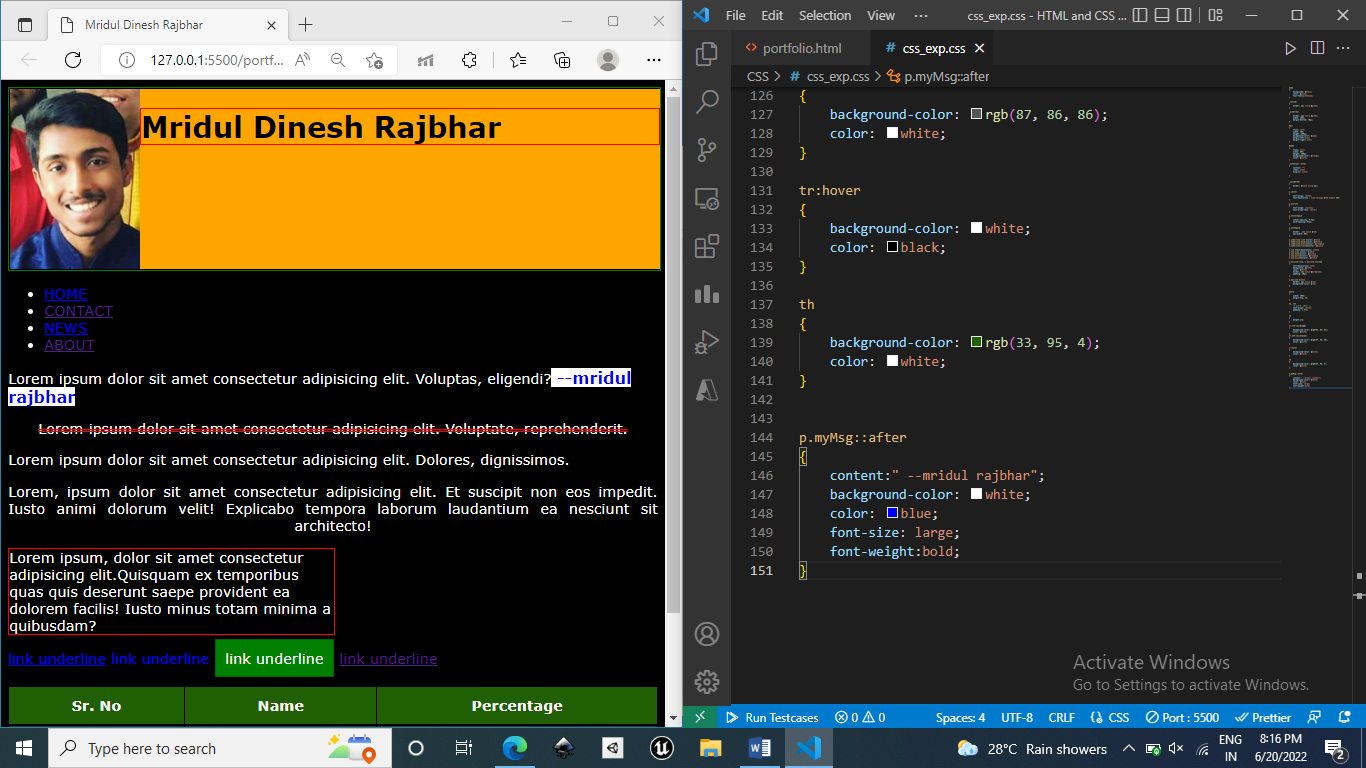
Syntax:

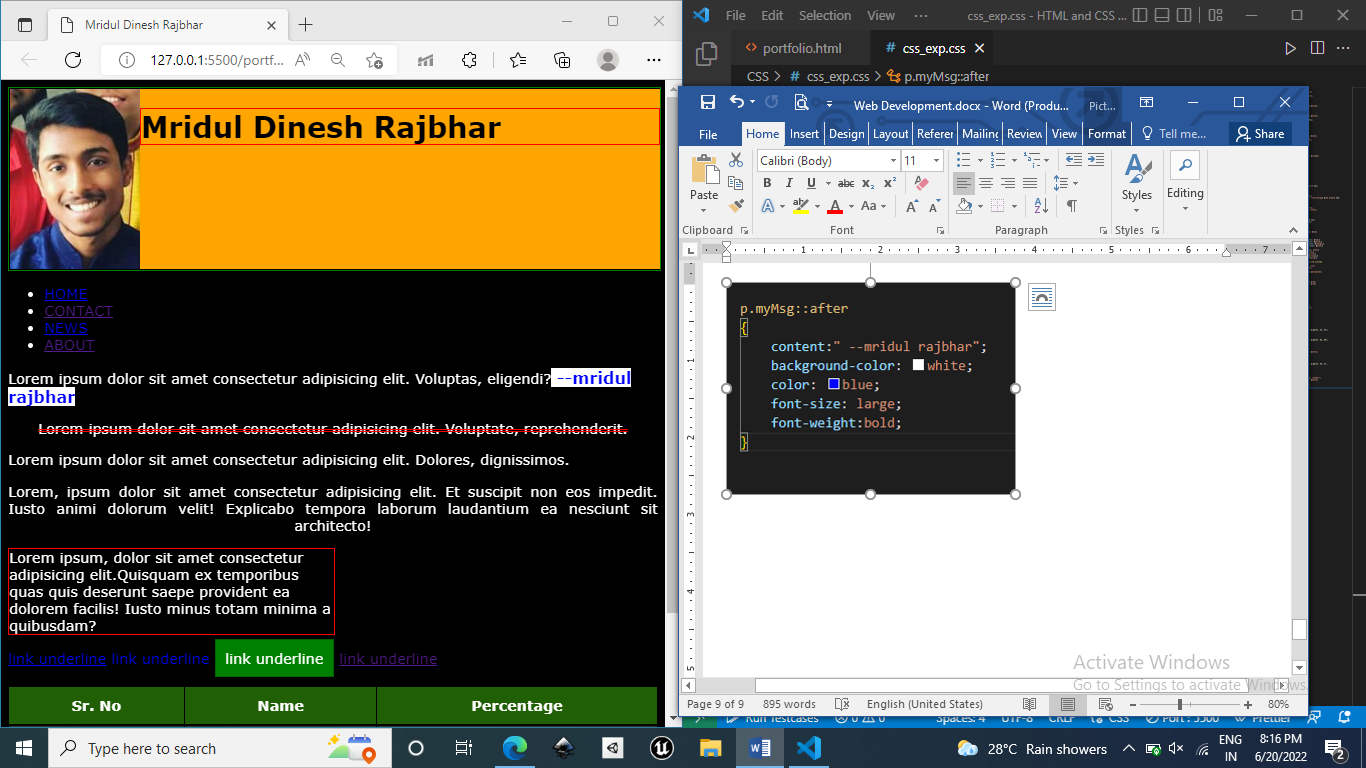
Selector::class

{property: value;}

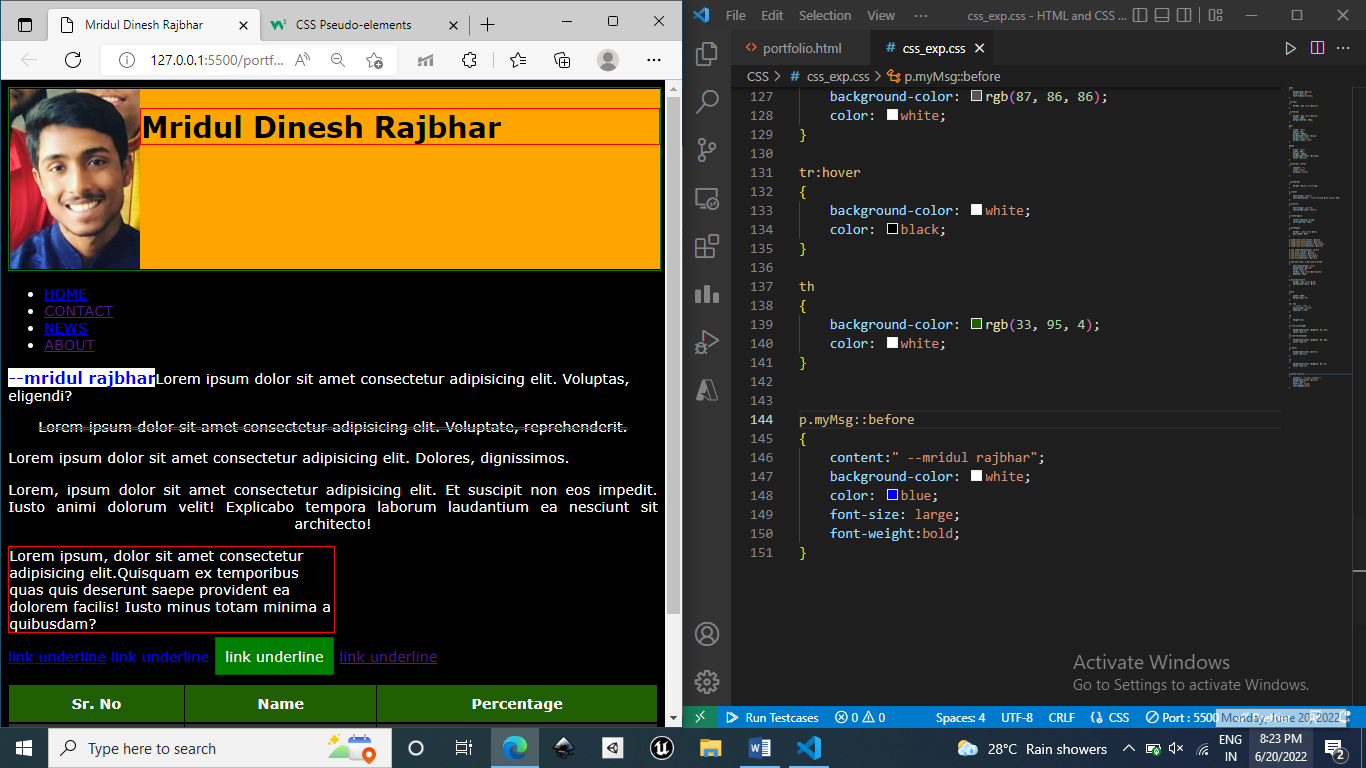
e.g

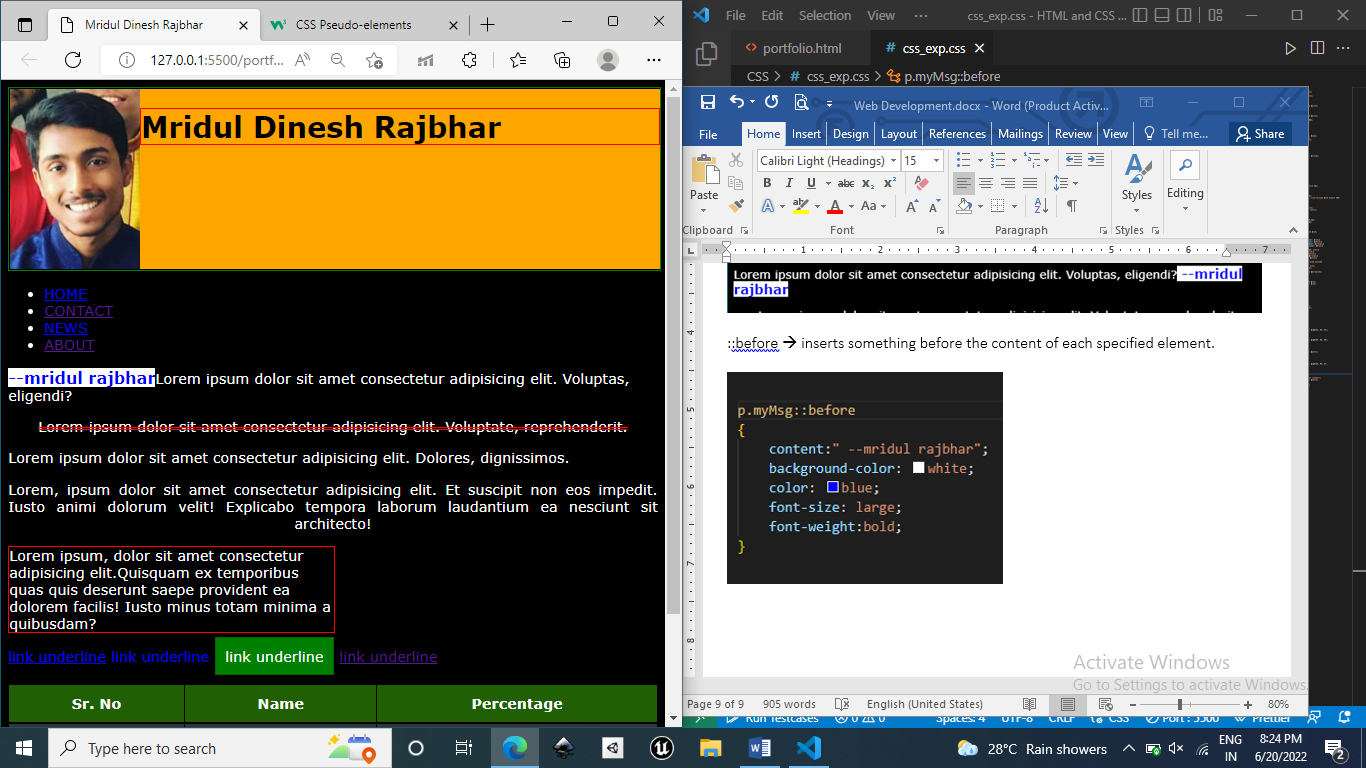
p::after 🡪 Insert something after the content of each p element.





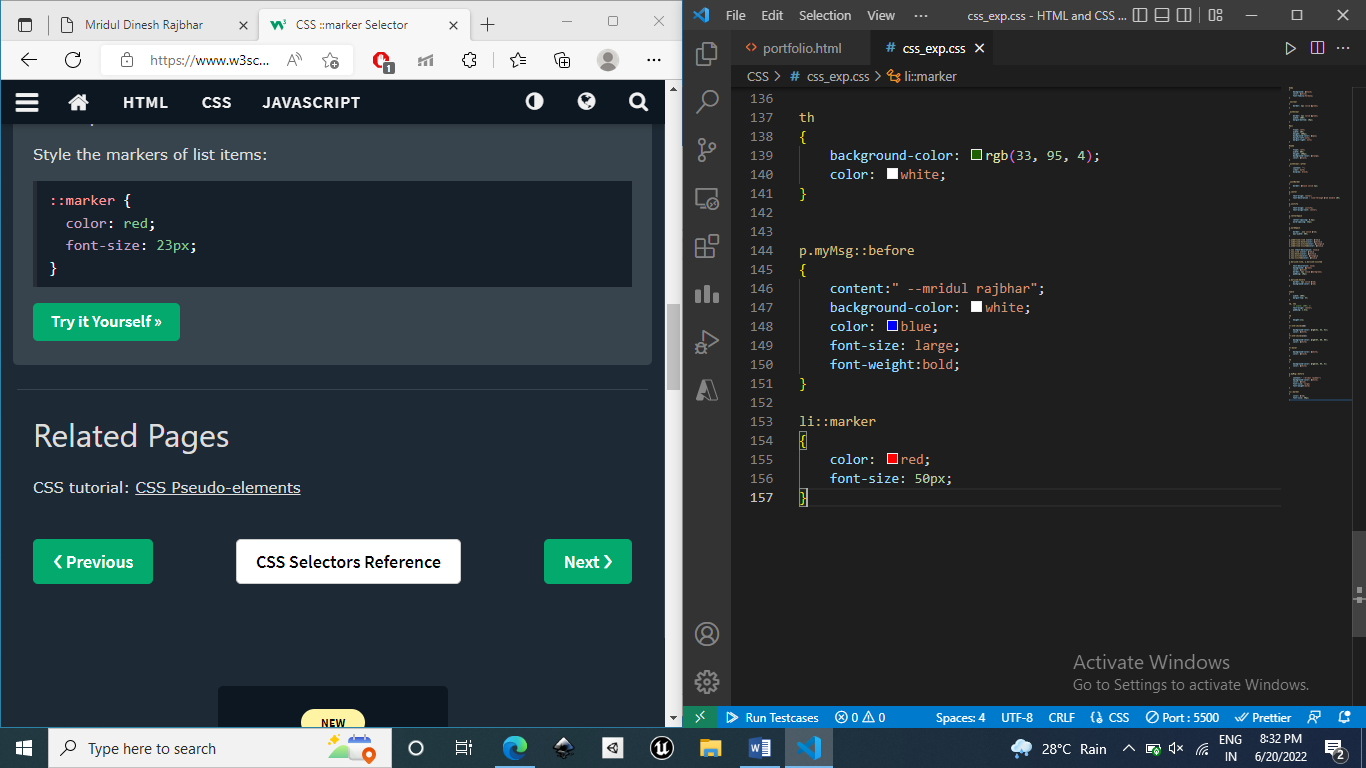
::before 🡪 inserts something before the content of each specified element.

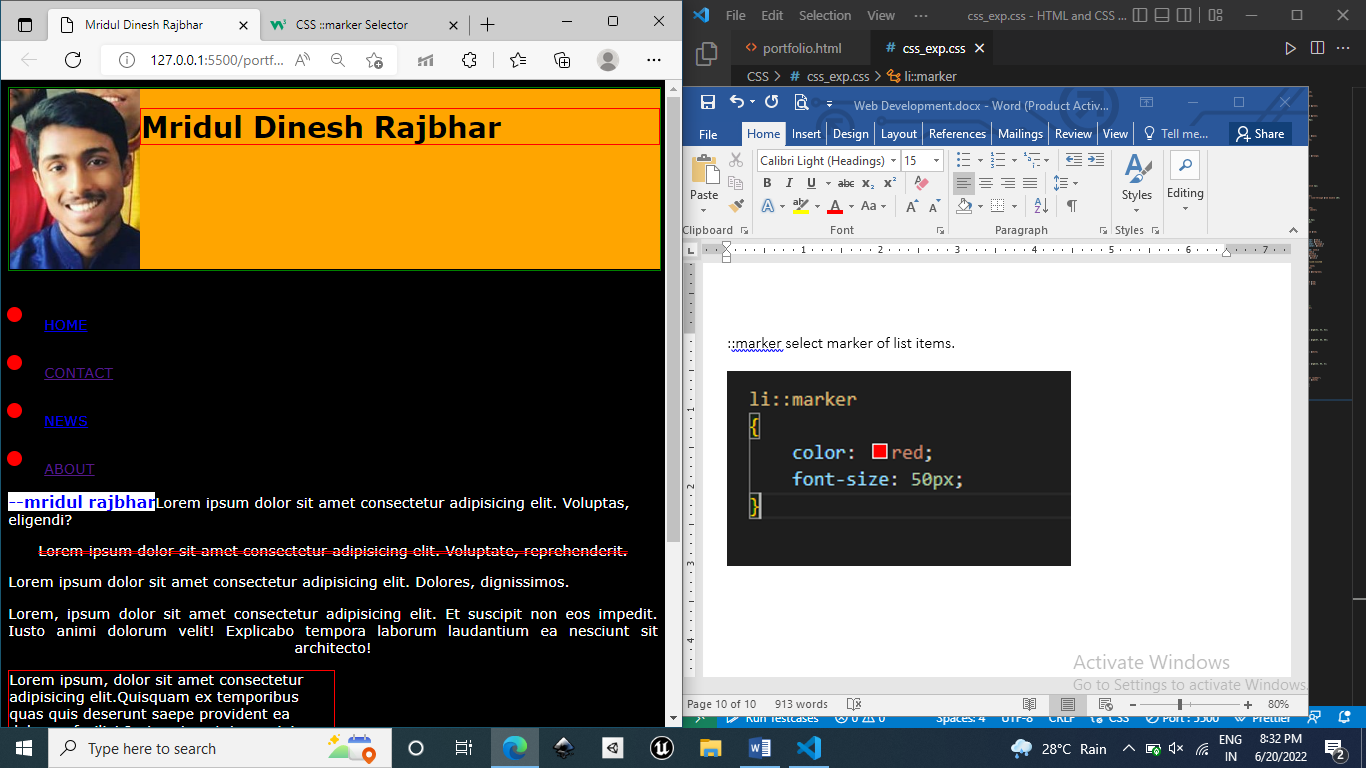




::first-letter, ::fisrt-letter

::marker select marker of list items.





::selection refers to any portion of the element that is selected by user.

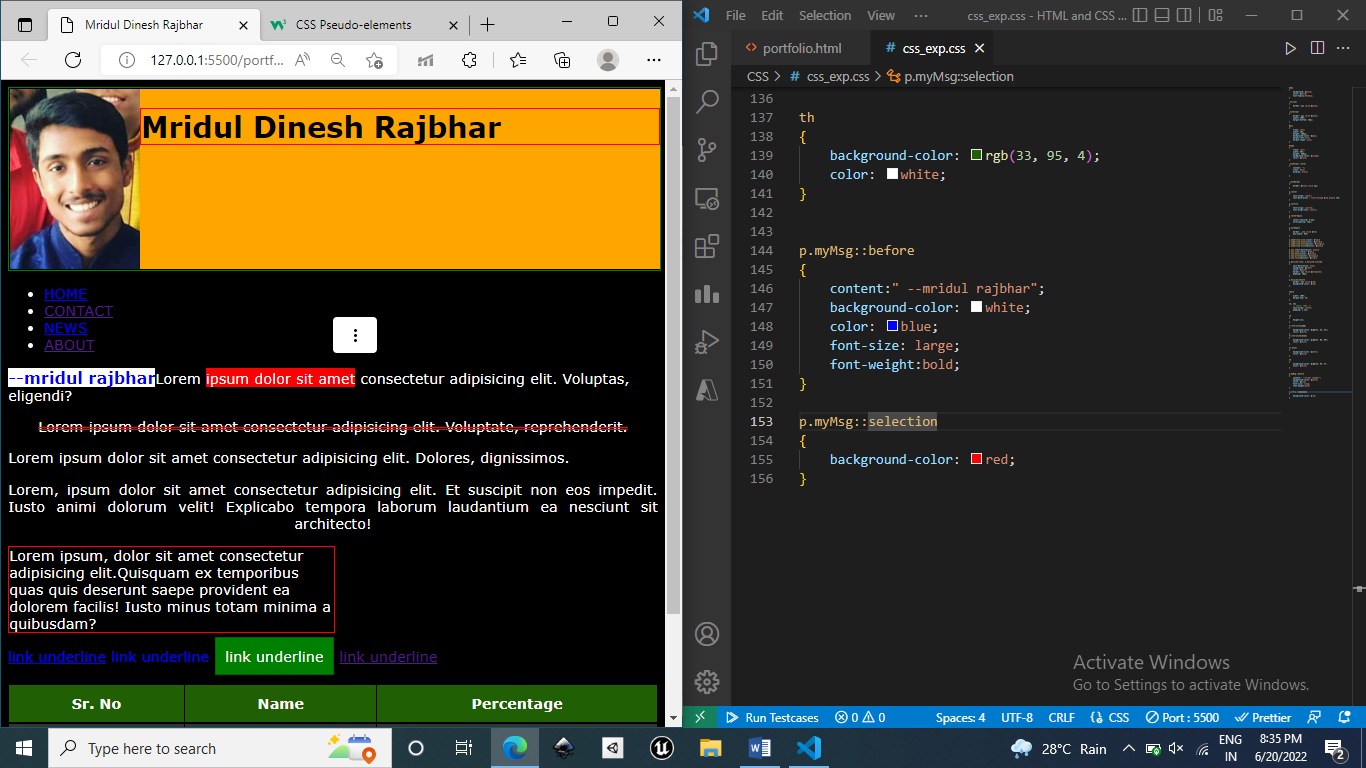
E.G

p.myMsg::selection

{

    background-color: red;

}

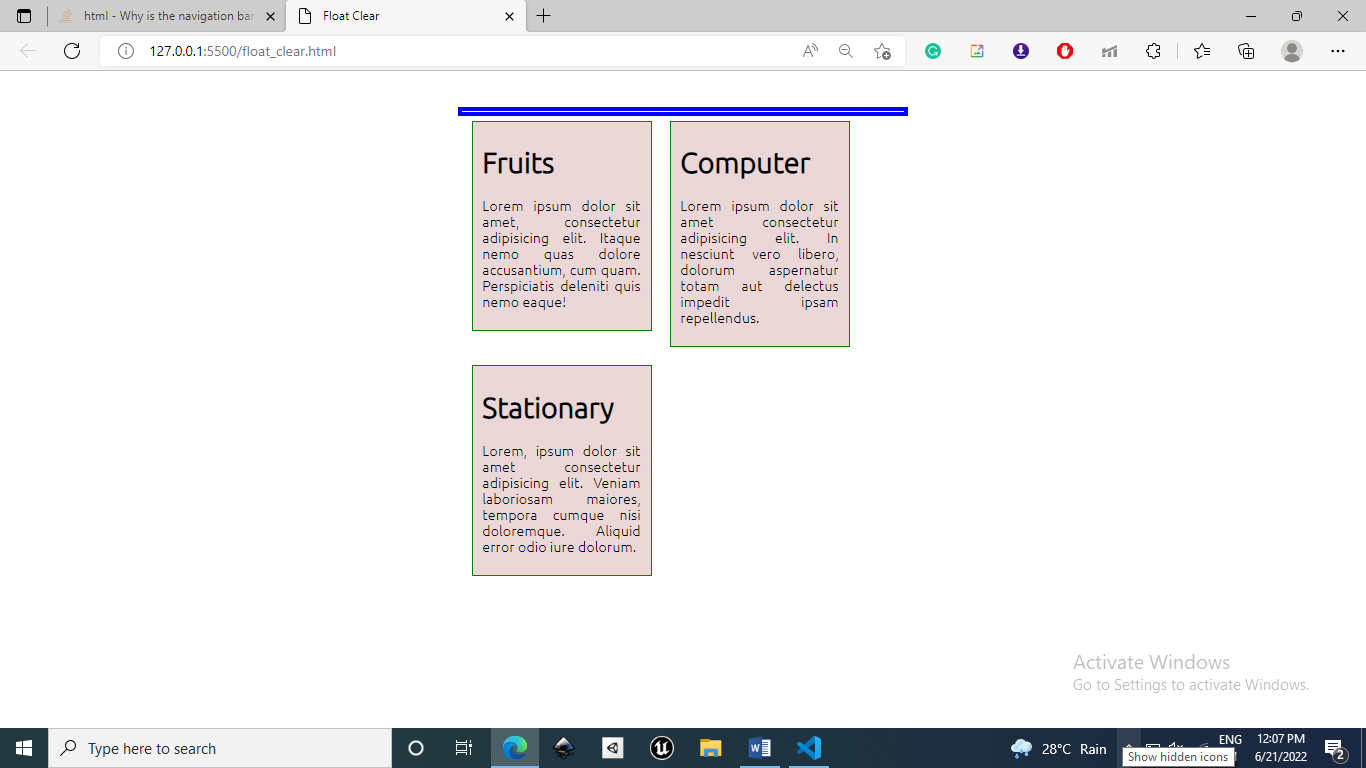


ClearFix Method

**Without clearfix**

The float element’s <li> height is not considered by the parent element ul and parent element is not expanding to include all the float elements.

So using clearfix method to solve the problem.



**With clearfix**

.clear-fix::after

{

    content: "";

    clear:both;

    display: block;

}

::after 🡪 Insert something after the content of element.

content: “” 🡪The content we want to insert.

Clear: both 🡪 if we want particular element to be below the float left and right elements.

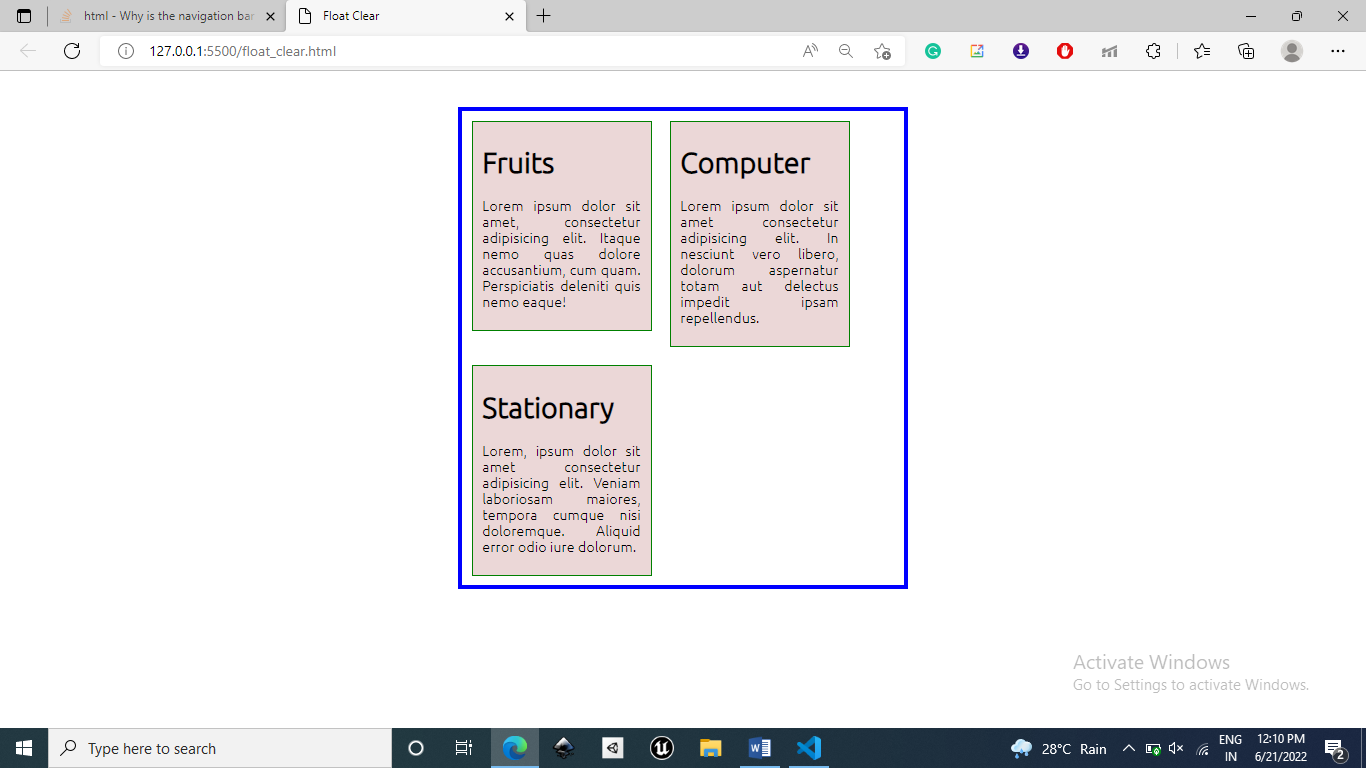


Image Sprites

An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

**Attribute Selectors**

**Syntax: element[attribute] / [attribute]**

e.g

a[target] 🡪 select all the <a> elements having target attributes.

Syntax: element[attribute = “value”]

a[target=”\_blank”] 🡪 select all the <a> elements having attribute target with value “\_blank”.

Syntax: [attribute~=value]

Select all the elements with the attribute value containing specified word.

Syntax: [attribute|=value]

Select all the elements with the attribute value having exactly same value or the value followed with hyphen(-)

E,g [class|=”top”]

<a class=”top right”> & <a class=”top”> both the element will be selected.

Syntax: [class^=top]

If the first value of the specified attribute is having specified value then it will be selected.

Syntax: [class$=”top”]

If the last value of the specified attribute is having specified value then it will be selected.

Syntax: [class\*=”te”]

If the value of the specified attribute is having the specified substring then it will be selected.

e.g

[class=”te”]

<a class=”ten”> , <a class=”teen”>, <a class=”te”> will be selected.

**FORMS**

**<label>**

Adding labels to the input elements.

<label for=”name”>Name:</label>

<input type=”text” id =”name”>

\*\*\*the value of for attribute of the label must be same as value of the element id so that label becomes associated with that input element.

Notice the use of the <label> element in the example above.

The <label> tag defines a label for many form elements.

The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

The <label> element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.

The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

**Submitting the form**

<input=”submit”> is used to submit the form data to the form handler(script on the server). Form handler is processing the input data. The form handler is specified at action attribute.

If there will be no name attribute in input field then its value won’t be submitted(Its value won’t be send to server when submitting).

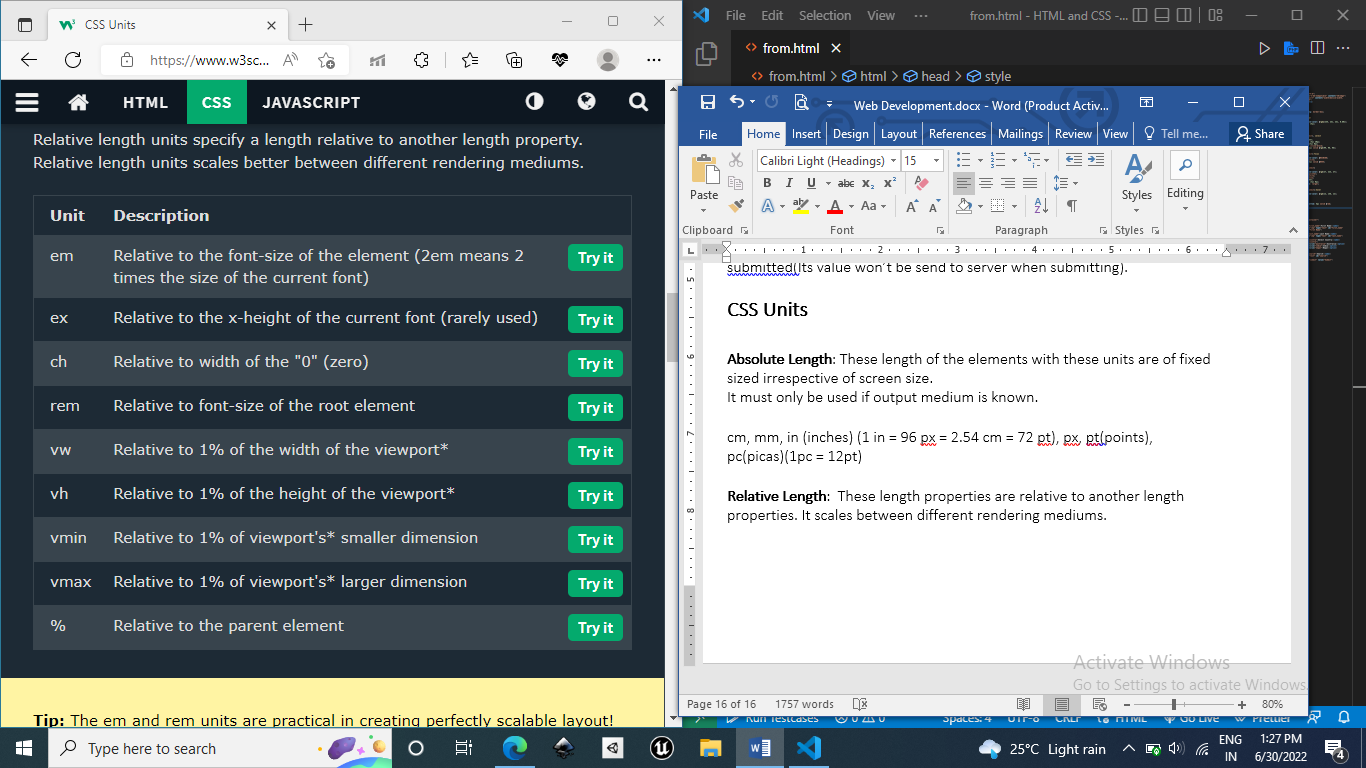
**CSS Units**

**Absolute Length**: These length of the elements with these units are of fixed sized irrespective of screen size.

It must only be used if output medium is known.

cm, mm, in (inches) (1 in = 96 px = 2.54 cm = 72 pt), px, pt(points), pc(picas)(1pc = 12pt)

**Relative Length**: These length properties are relative to another length properties. It scales between different rendering mediums.



Specificity(can be said as priority)

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

Four Categories of specificity and there order of importance in descending order:

1. Inline styles (1000 pts)

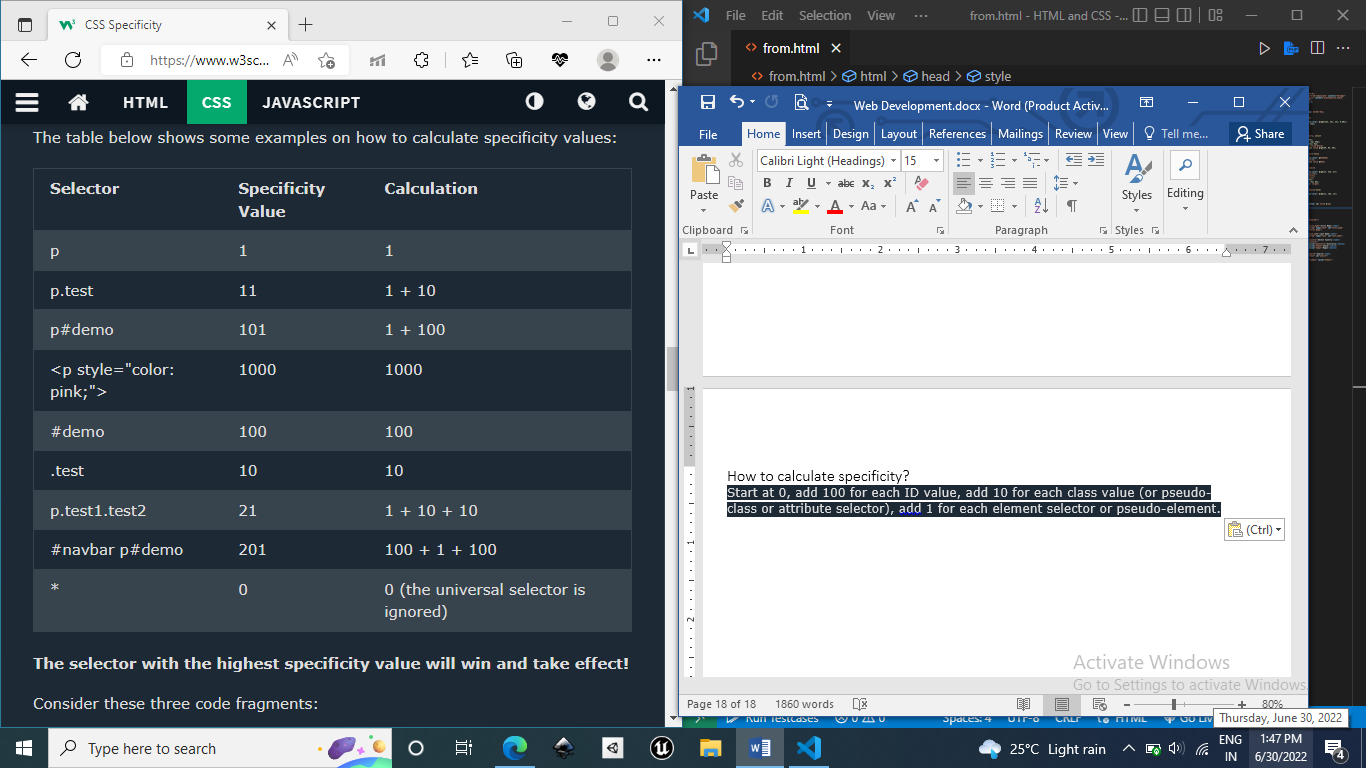
2. IDs (100 pts)

3. Classes, pseudo-classes, attribute selectors (.test, :hover, [href])(10pts)

4. Elements and pseudo-elements (h1, p, h1::after)(1pt)

How to calculate specificity?

Start at 0, add 100 for each ID value, add 10 for each class value (or pseudo-class or attribute selector), add 1 for each element selector or pseudo-element.



**Equal Specificity rule: latest rule wins**

If same rule is written twice the latest rule in external style sheet wins.

h1{color: red}

h1{color: yellow}

so the second rule will be applied.

**Specificity on basis of files**

Inline CSS(high), Internal CSS(Mid), External CSS(Low)

**!important**

It will give more importance to a specific property/value than normal and will override all the styling for the specific property of that element.

#myid {  
  background-color: blue;  
}  
.myclass {  
  background-color: gray;  
}  
p {  
  background-color: red **!important**;  
}

All the paragraphs will have red color because !important rule has overridden all the previous background property.

**Math Functions**

Calc(), min(), max()

Calc() 🡪 to perform calculation to find value of css property.

Width: calc(100% - 100px)

Max(50%, 300px, 4em) select largest value.

Min() 🡪 select smallest value.

**Border radius**

Border-radius: (top-left) (top-right) (bottom-right) (bottom-left)

Border-radius: (top-left) (top-right/bottom-left) (bottom-right)

Border-radius: (top-left/bottom-right) (top-right/bottom-left)

**Border Images**

It allows us to use the image as the border around the element.

Syntax:

Border-image has 3 properties:

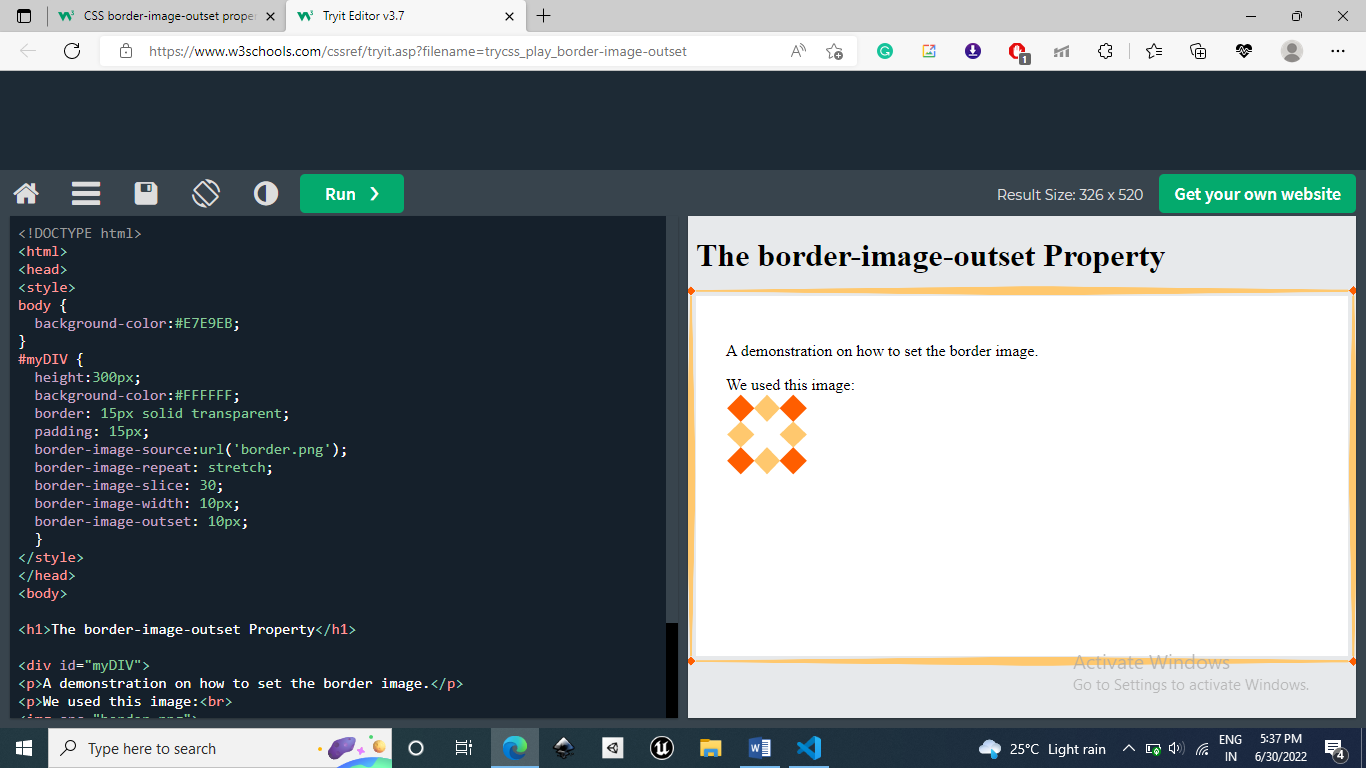
1. image to be used
2. where to slice the image
3. middle section must be repeated or stretched.

Image is divided into 9 sections 4 edges, 4 corners, 1 middle section.

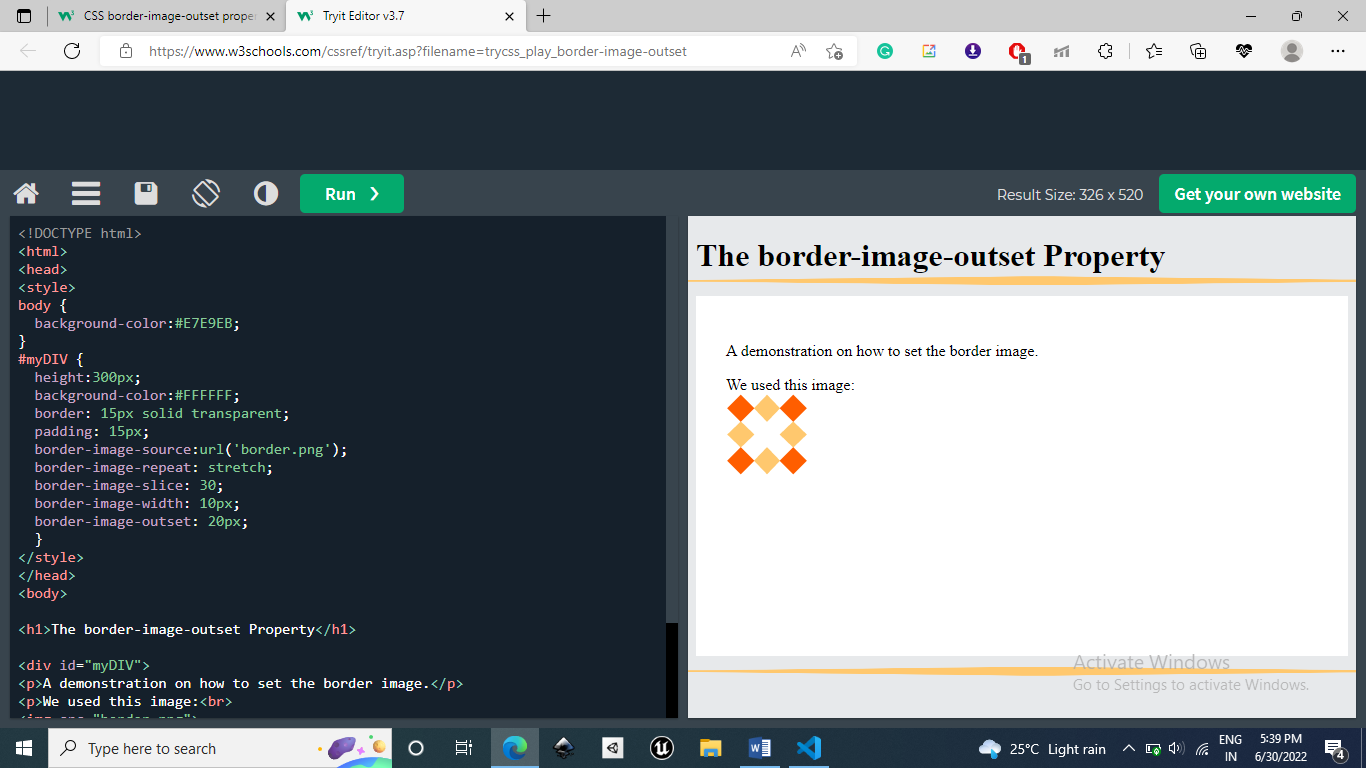
Border-image-outset: amount of area by which we have to extend our border image area form border box.

Border-image-box

border-image-box: 10px;



Border-image-box: 20px;



**Multiple Backgrounds**

Background-image property is used to have multiple images in the background by comma “,” separating.

Background-image: url(‘image1’), url(‘image2’);

The images are stacked over each other and the first image is closer to viewport. (Here image1 is more closer to image2).

Other properties are also set using comma “,” separation.

#example1 {  
  background-image: url(img\_flwr.gif), url(paper.gif);  
  background-position: right bottom, left top;  
  background-repeat: no-repeat, repeat;  
}

Background is the shorthand property.

Background: image position repeat.

#example1 {  
  background: url(img\_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;  
}

**Color Keywords**

Transparent 🡪 It is equivalent to rgba(0,0,0,0).

**Cover and Contain**

cover tells the browser to make sure the image always covers the entire container, even if it has to stretch the image or cut a little bit off one of the edges.

contain, on the other hand, says to always show the whole image, even if that leaves a little space to the sides or bottom

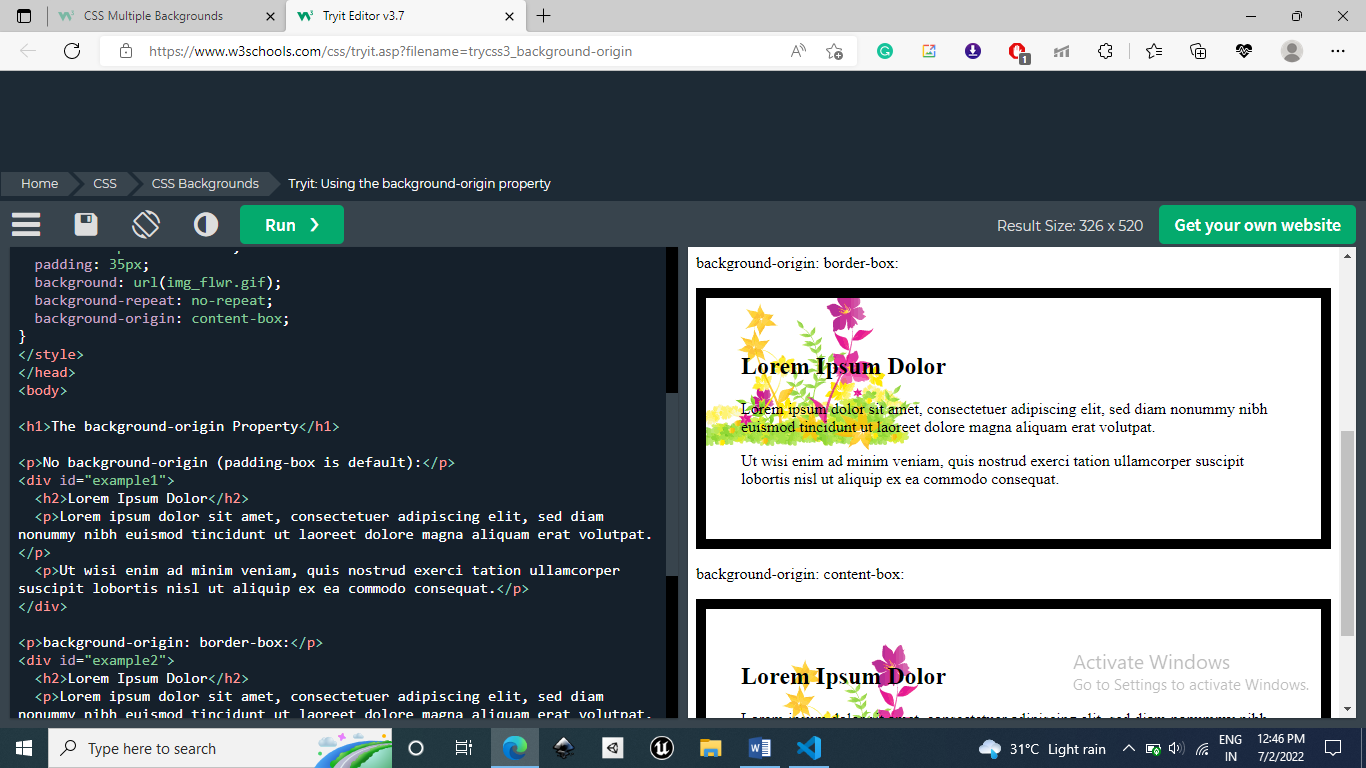
**define multiple size of background**

background: url(image1), url(image2);

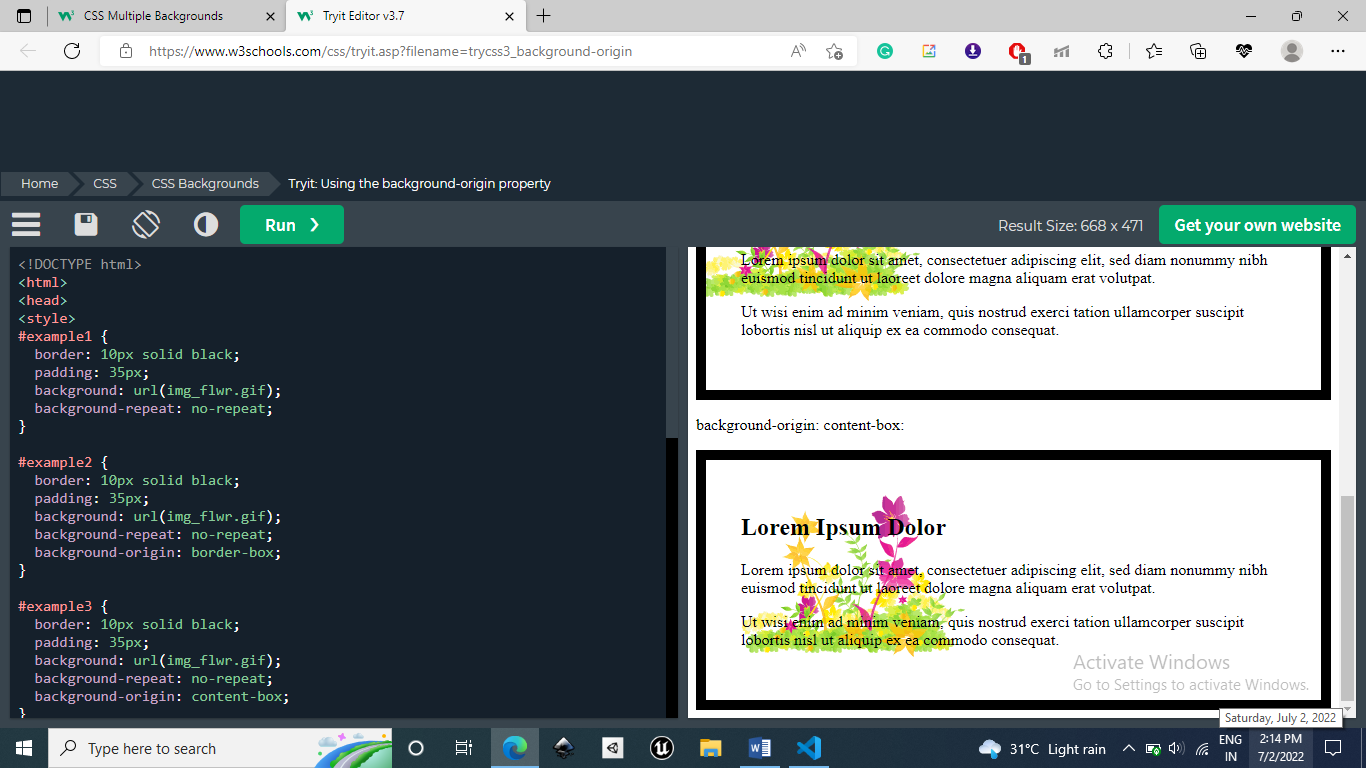
background-size: size\_of\_image1, size\_of\_image2;

**Background-origin**

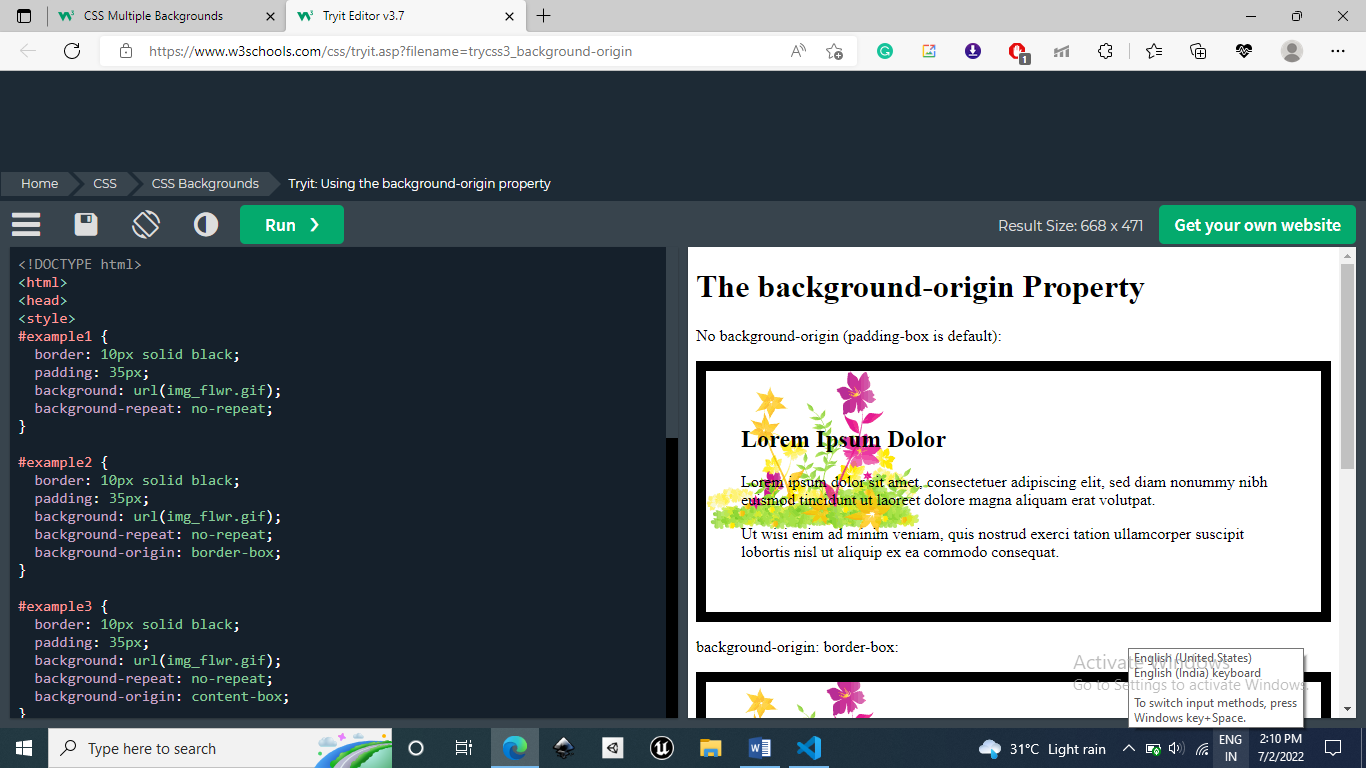
1. border-box : Starts from upper-left corner of the border.



1. content-box: Starts positioning of bg from upper-left of content.



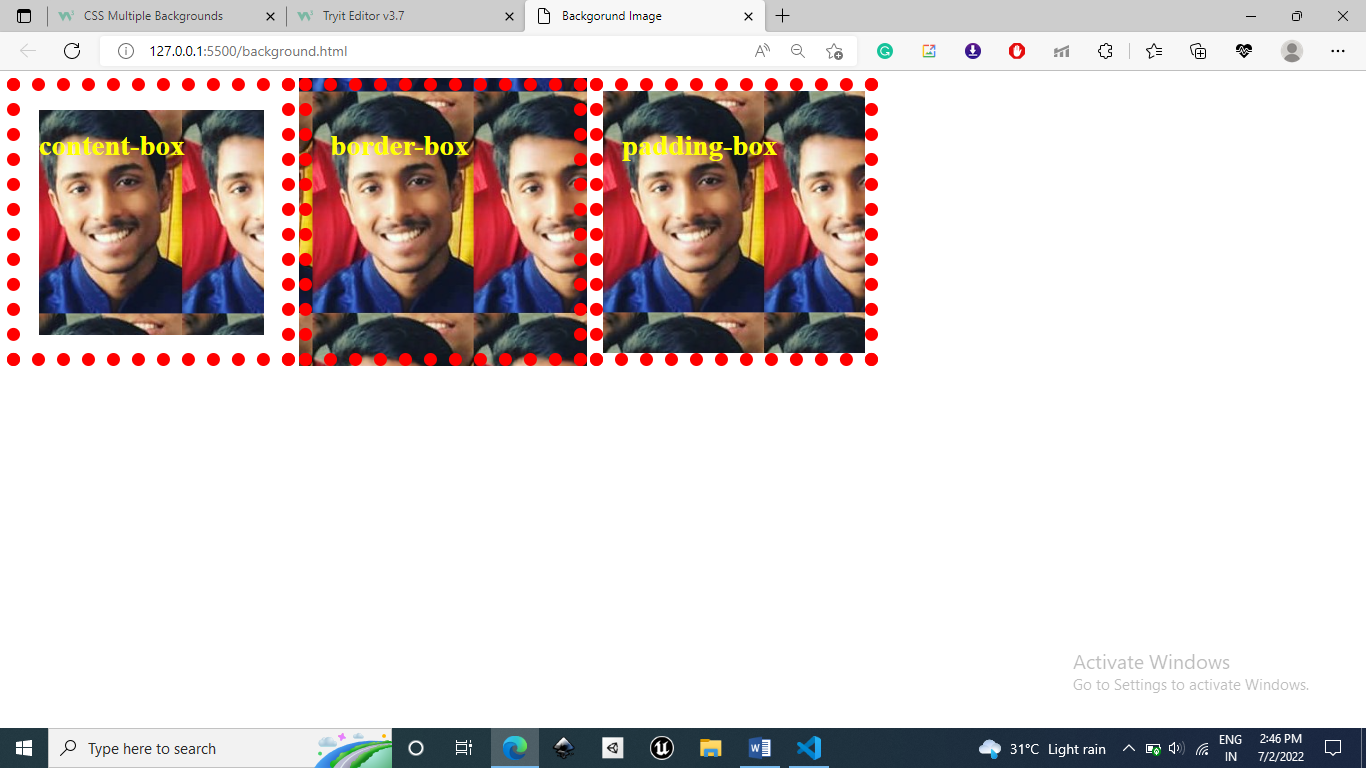
1. content-box: Starts positioning of bg from upper-left of padding.



**Background-clip:**

How much background area must be painted.

* border-box - (default) the background is painted to the outside edge of the border
* padding-box - the background is painted to the outside edge of the padding
* content-box - the background is painted within the content box.



**Color**

An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

**HSL**

HSL stands for Hue, Saturation and Lightness.

An HSL color value is specified with: hsl(hue, saturation, lightness).

1. Hue is a degree on the color wheel (from 0 to 360):
   * 0 (or 360) is red
   * 120 is green
   * 240 is blue
2. Saturation is a percentage value: 100% is the full color.
3. Lightness is also a percentage; 0% is dark (black) and 100% is white.

**HSLA (A 🡪 Alpha(transparancy) )**

**Opacity**

The CSS opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent).

**Color Keywords**

transparent 🡪 equivalent to rgba(0,0,0,0).

Currentcolor 🡪 It is a variable which stores the current value of the color property.

**Text-shadow** 🡪 to set shadow of the text.

text-shadow: horizontal\_displacement vertical\_displacement blur\_effect color.

Use comma separation to add multiple backgrounds.

text-shadow: horizontal\_displacement vertical\_displacement blur\_effect color, horizontal\_displacement vertical\_displacement blur\_effect color;

**Spread parameter**

It defines spread radius of the shadow. If positive value then shadow size increases and if negative then size decreases.

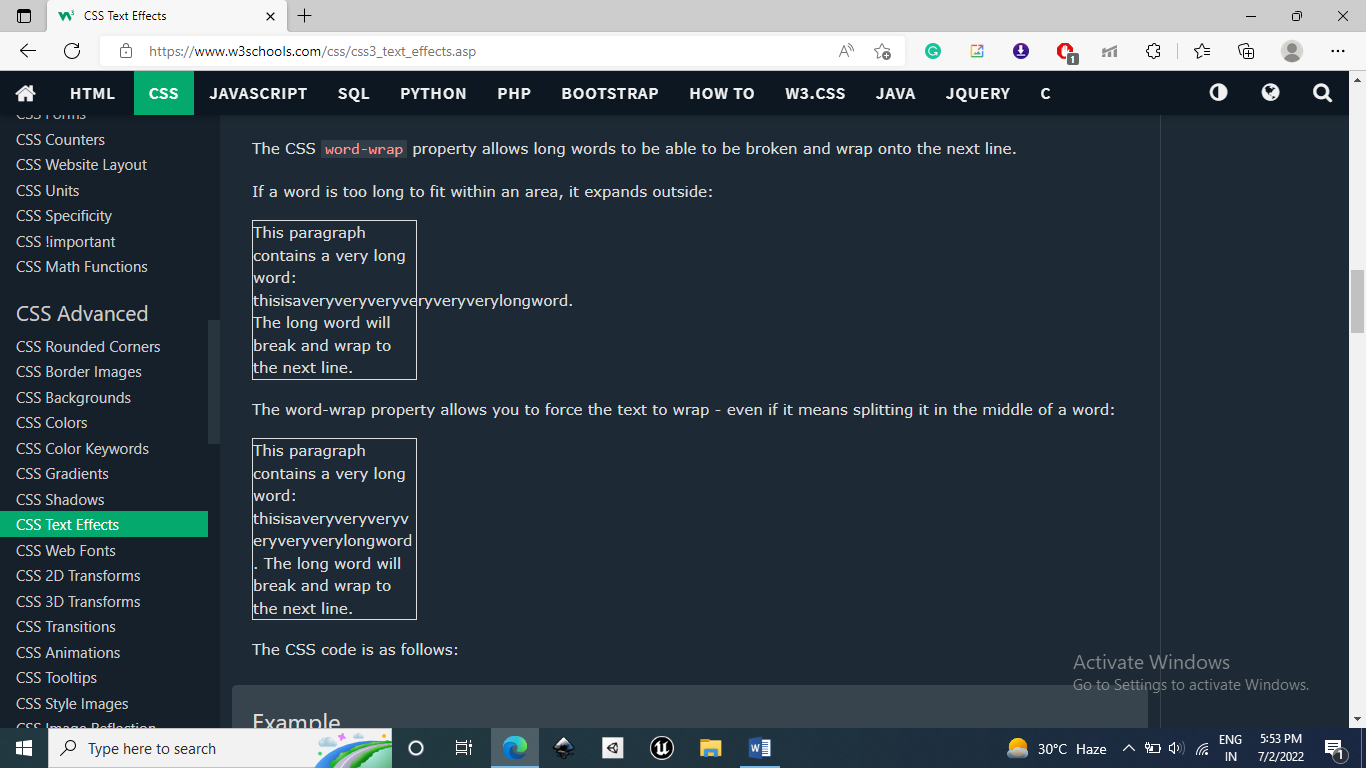
**Inset Parameter**

Inset parameter changes shadow from an outer shadow to inner shadow(shadow at left and top side).

**Text Effects**

Word-wrap 🡪 breaks the word and wraps it into the next line.

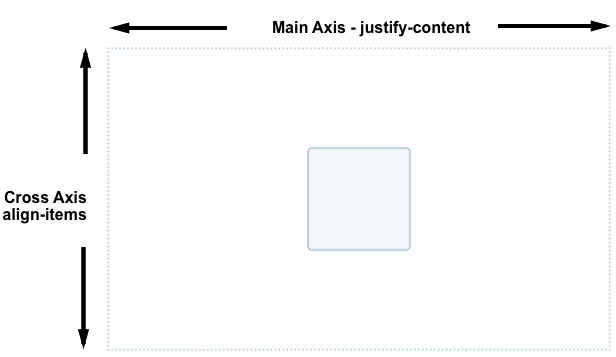
Break-word 🡪 allows unbreakable word to be broken.



p{word-wrap: break-word;}

Ideally there should be only one h1 heading in the page for search engines like google and bing to know what is important heading in your image.

**Justify Content v/s Align Items**



Topics to Understand

Border image Slicing