HTML Basics:

Head elements:

<meta> --- not shows in page but used to define keywords and descriptions of webpage

 <meta charset="utf-8">

    <!-- define keywords for search engine -->

    <meta name="keywords"  content="HTML,CSS"></meta>

    <!--description of website-->

    <meta name="description" content="This is a website for learning HTML and CSS"></meta>

    <!-- define author of website -->

    <meta name="author" content="Then"></meta>

    <!-- define viewport for all  device to look site look good on all device-->

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

* The <meta> element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings

1. **Charset** - The charset attribute specifies the character encoding for the HTML document(preferred UTF-8 – covers almost all character and numbers in world).
2. **Content** – specifies values associated with name or other attribute
3. **Name** – specifies name of metatag
   1. **Viewport -** viewport for all device to look site look good on all device .. **initial-scale** - define inital zoom level when page loaded for first time,
   2. Content – width = device-width – set default width of content

Without viewport:

A screenshot of a computer

Description automatically generated

With viewport:

A screenshot of a computer

Description automatically generated

**HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag**.

You should include the following <meta> viewport element in all your web pages:

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

This gives the browser instructions on how to control the page's dimensions and scaling.

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.

Size Content to The Viewport

Users are used to scroll websites vertically on both desktop and mobile devices - but not horizontally!

So, if the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience.

Some additional rules to follow:

**1. Do NOT use large fixed width elements -**For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally. Remember to adjust this content to fit within the width of the viewport.

**2. Do NOT let the content rely on a particular viewport width to render well** - Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.

**3. Use CSS media queries to apply different styling for small and large screens** - Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport on a smaller device. Instead, consider using relative width values, such as width: 100%. **Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.**

**CSS – cascading style sheet:**

**Syntax:**

Selector{property : value} => h1{font-color:red}

**Types of css:**

* Inline – 1st priority
* Internal -2nd
* External -3rd

**Selector:**

* Element
* Id
* Class

**Grouping tags in html :**

* **Div –**used to group multiple element together
* **Span –** used to group inline content

**Inheritance :**

* Child element inherit style from parent, unless the child has its own style
* If direct child have any style property then it will not inherit its parent property
* Only if the property Is overridden then it will effect

**Specificity :**

* When there is one property present for a single element multiple times then specificity is factor that decides which property should be taken into consideration
* The property with high specificity will take precedence and will be applied to element

**1.Universal selector and inherit property = 0 specificity**

2.!important – has more specificity than inline style

3.**Id = 100**– high

4.**class, pseudo-classes, attribute selectors** = 10 – medium

5.**elements(div,p) and pseudo elements ::before** – least -1

**Note :** If both elements has same specificity then it will follow last row principle i.e the property defined at last will take precedence

Colors in css:

Background : red;

**Background:rgb(25,0,0,0.5); -- last parm – opacity**

**Decimal : #FF0000**

/\* hexadecimal \*/

/\* #RRGGBB \*/

/\* 123456789  A(10) B(11) C(12) D(13) E(14) F(15)

#FF0000 - red

#00FF00 - green

#0000FF - blue

Colors: coolers.co

**Css units:**

* **PX –** it is absolute unit which represents px of screen . It wont change based on screen size
* **% - it is relative unit** – which means it depends on it parents property .. its value is calculated based on its parent value .. when there is no parent present for element then browser value will be considered.

/\* .outer

{

    width:200px;

    height: 300px;

    background:blue;

}

.inner

{ \*/

    /\* this means it takes 100% width of 200px i.e parent \*/

    /\* width:100%;

    height:67%;

    background: red;

} \*/

<div class="outer">

    <div class="inner">

    </div>

</div>

In above example the width present is calculated based on outer class width

* **Em – relative unit –** this unit depends on base value of parent element or if no parent element is present then browser size is taken for calculation
  + **Base value\*em value =** eg : 16(base size of browser \*2em) = 32px

/\* Em - relative they depend on parent - em is calculeated based on basevalue\*em value 16\*2  eg: if we change the font of

 browser then the em value will be calculated based on font size\*rem if there is no base element\*/

 /\* if there is any base element present then the em value will be 10\*2=20px \*/

 /\* they are going to depend on parent if no parent present then actual browser value will be used for calculated \*/

div{

    font-size: 10px;

}

.absolute

{

font-size: 32px;

}

.relative{

    /\* 0.5em is similer to 5px  \*/

font-size: 2em;

}

* **Viewport unit -vh,vm**

**viewPort units -vh,vw -relative units**

vh - height - percent of screen - depend on screen

vw - width - percent of screen - same for width - percent from screen

regardless of screen size the value is calculated based on screen size

relative to screen

.hero

{

    background: green;

    width:100vw;

    height:100vh;

}

This overs 100 percent of screen size

**Default browser style:**

default browser tools

* there is some default browser -- default styling to show the element in browser. So, there are some base styles present in browser
* we can override the default styles.

**Calc function:**

ul{

    background: blue;

    height:20rem;

    color:white;

    font-size: 3rem;

}

.banner{

    background: red;

    height:calc(100vh - 20rem);  /\* thre should always be a space b/t operation \*/

}

To calculate the height based on previous value or any other value we can use calc

Calc(100px + 2rem) or calc(3px + 3px)

**Min :**

A red and blue rectangle on a computer screen

Description automatically generated

.example

{

    background:green;

    width:10rem;

  /\*  height:auto;  it will take the height of content or any element inside the main element - \*/

height:15rem;

overflow: hidden;

overflow: scroll;

}

**Overflow** : it Is going to be used to hide the content that exceed given height or provide scroll for more content

**Height : auto** – height is going to vary based on content given inside the div for which height is given

When no content given inside the div then the element will not be show

**Min-height** : we are going to set min height for div to solve the issue related to when there is more content to div than height mentioned then in this case we can set min-height for element

**Max-height:**set max height for element

**TYPOGRAPHY:**

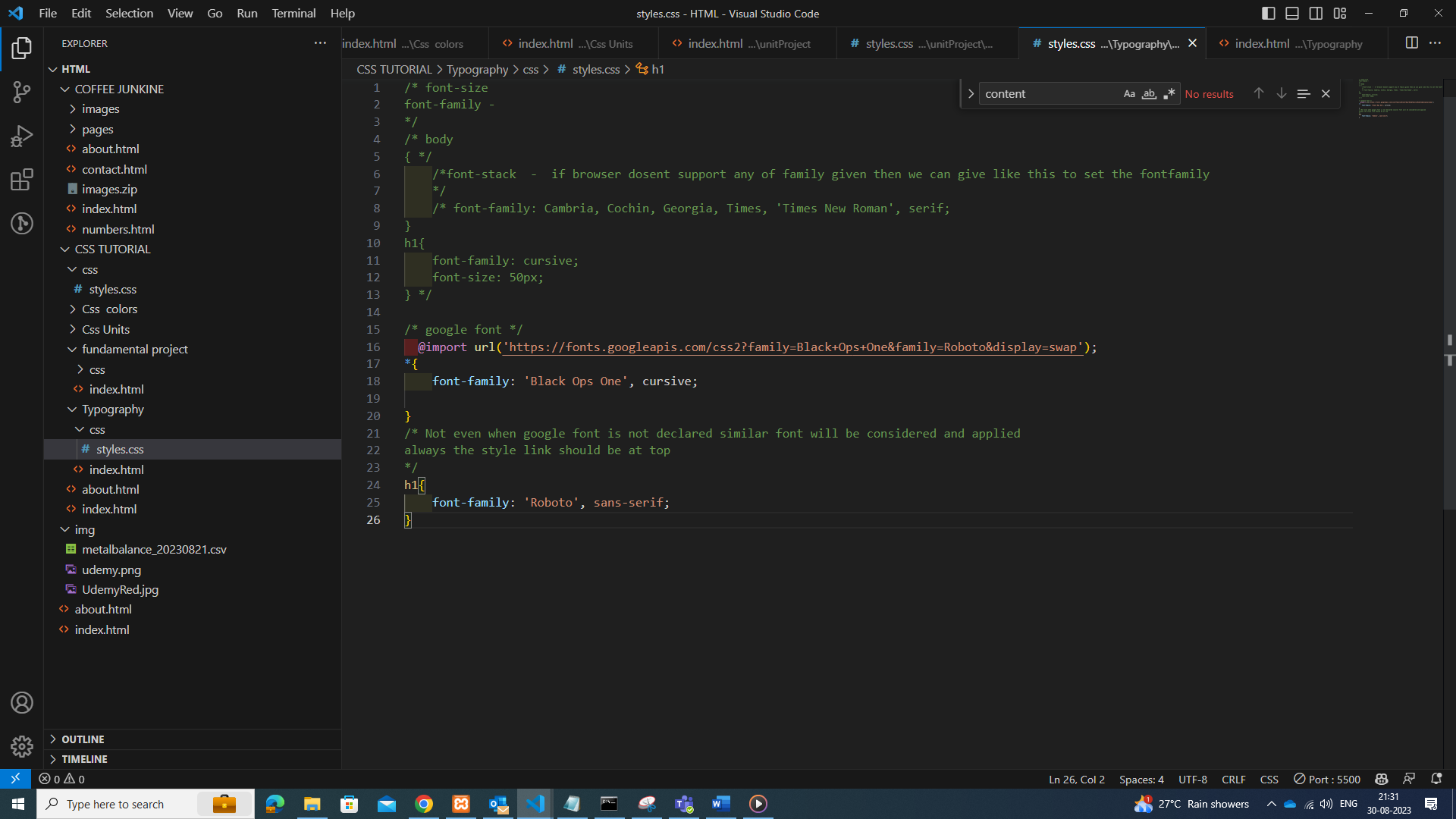
FONTS:

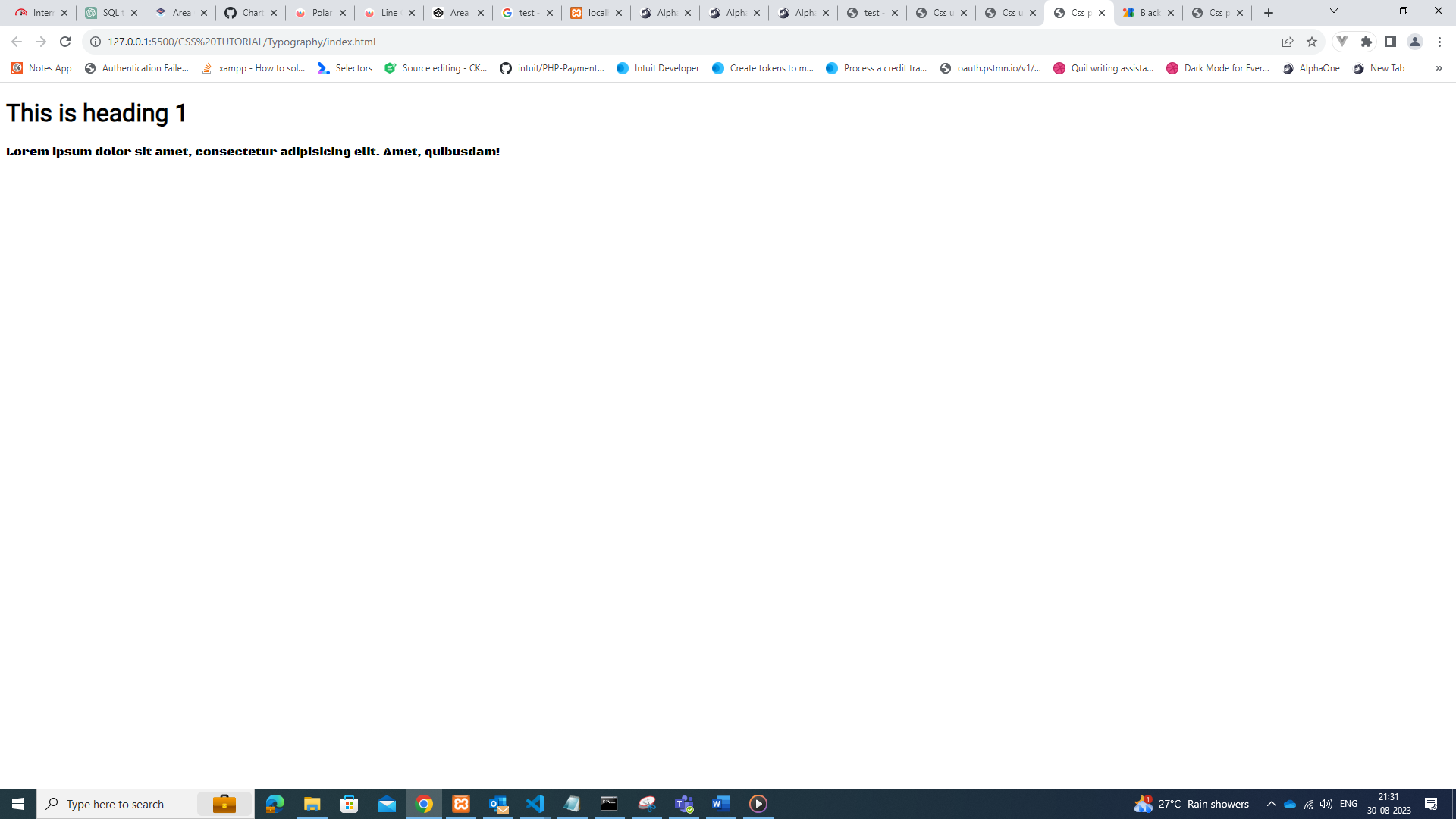
**Font-stack –** When browser doesn’t support any of font given then it takes the last one in list and apply it

Eg:  font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;

If non of above matches then **serif** will be taken as font family

Google fonts:





<https://fonts.google.com/specimen/Black+Ops+One?preview.size=41>

difference b/t rem and em :

* Rem depending on font size or any property of root element - the unit is relative to the font size of its parent element
* Em depending on font size or any property of its parent element when no parent is present then it will consider browser elements property - The rem is based upon the font-size value of the root element, which is the <html> element. And if the <html> element doesn’t have a specified font-size, the browser default value of 16px is used. So here only the value of the root is considered, and there is no relation with a parent element.

| **Parameter** | **em** | **rem** |
| --- | --- | --- |
| **Relativity** | em is relative to the font-size of its direct or nearest parent | rem is relative to the [HTML](https://www.geeksforgeeks.org/html/) (root) font-size |
| **Compounding Effect** | This may lead to a compounding effect | Does not lead to a compounding effect |

**CSS box model:**

A picture containing text, screenshot, rectangle, font

Description automatically generatedbox size

**width : width + padding left + padding right +border left+border right**

**height : height + padding top + padding bottom + border top +border bottom should remember always**

**padding –** distance b/t content and edge of element

**Border –** around element

**Margin –** spacing b/t other elements or edge of screen

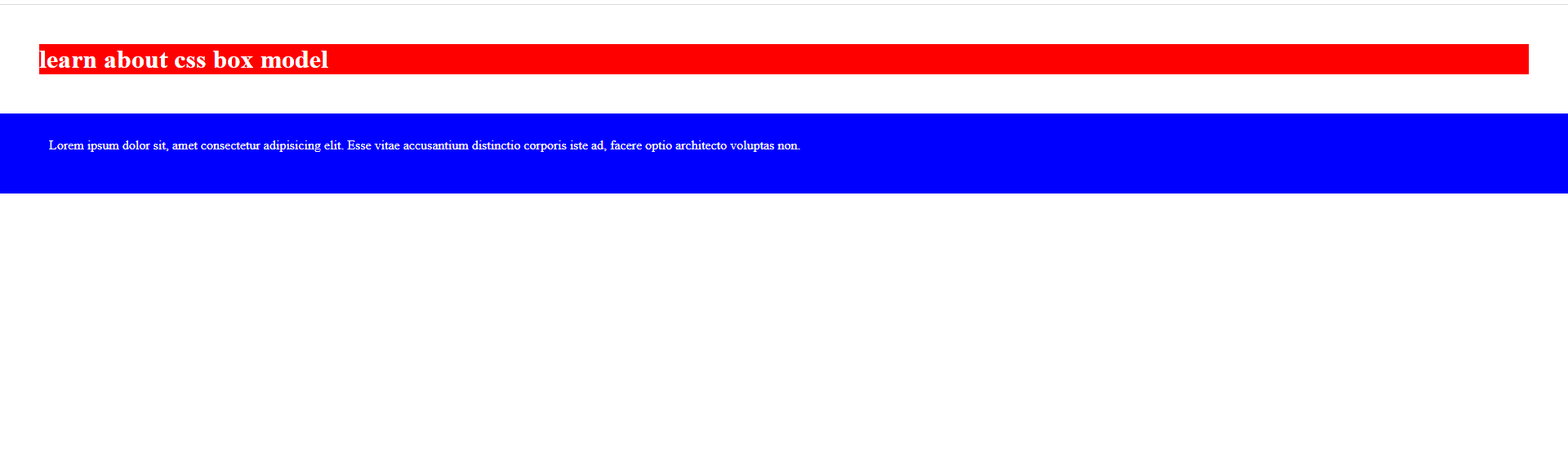
Padding :

**Padding shorthand** : padding: 30px 40px 50px 60px; /\* top right bottom left \*/

**Margin:**

Distance b/t 2 elements

**Margin shorthand** : margin : 30px 40px 50px 60px; /\* top right bottom left \*/



Distance b/t 2 elements – 2 elements are always going to be responsible for margin property

Border:

Outside of div

**Display property**

Element have set default display property

Block: spaning all accross. they will start new line always - always it will span whole width

inline:not going to start new line .. it will span only the content - only it will span to width of content

by default elements have display property

we can control the default behavior

links are inline element

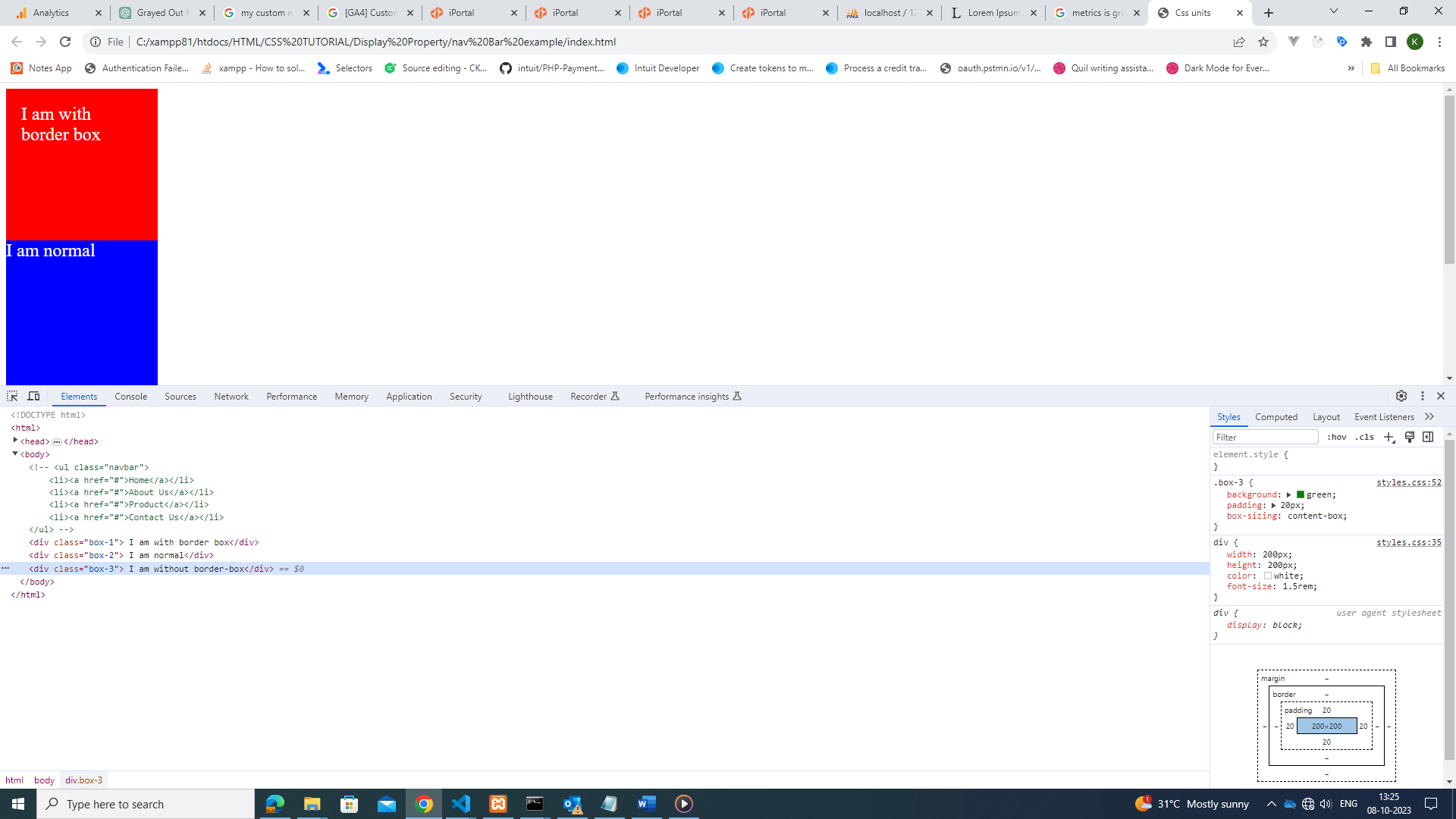
**browser will respect width,height top margin for block type element but for inline element browser wont respect width height**

If we need height width margin for inline element then we need to convert the inline element to block type element

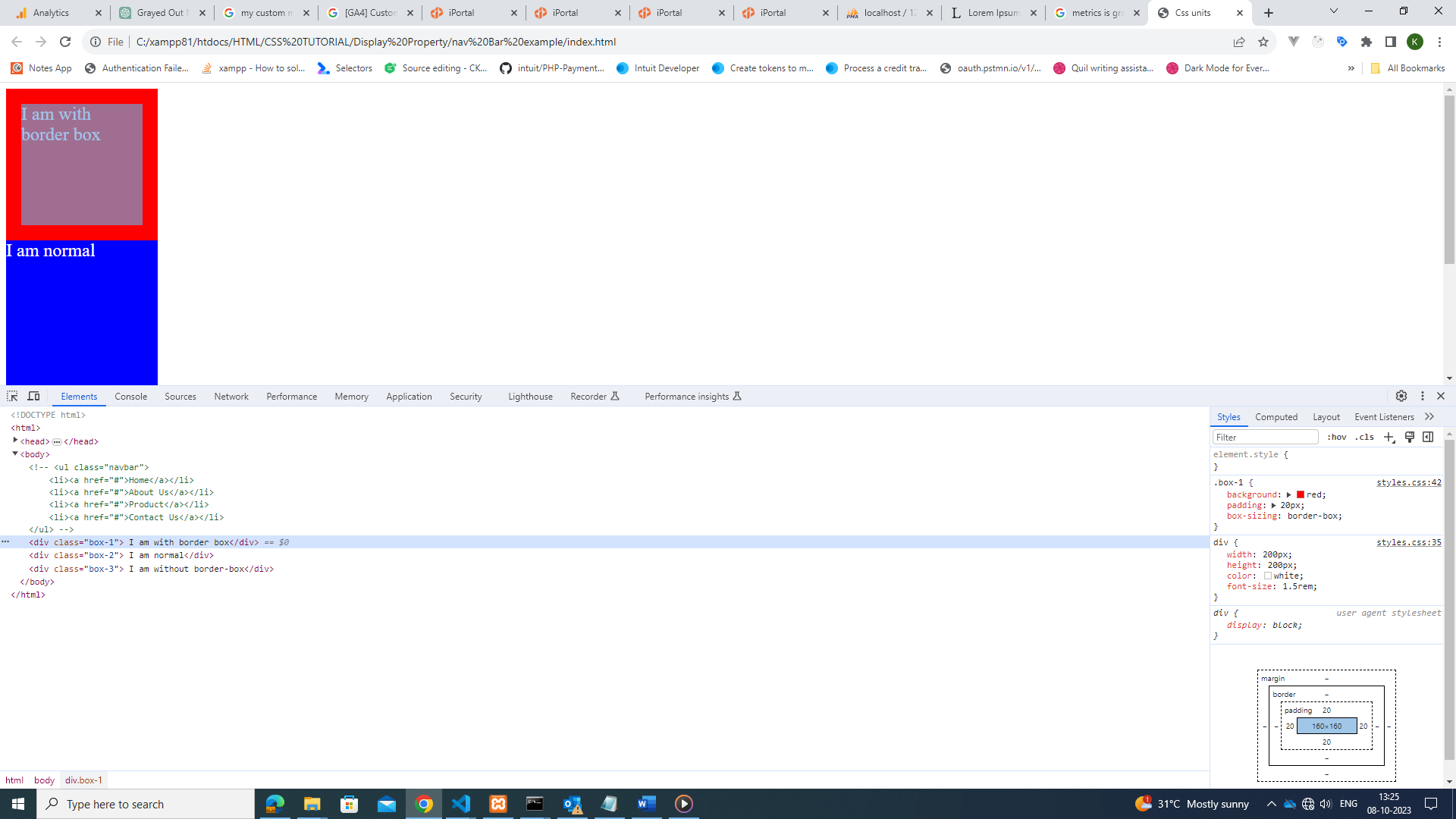
* Display : block – respect width , margin , height and padding and it will always begin in new line … eg: div
* Display : inline – don’t respect width , margin , height and it wont begin in new line
* Display : inline – block – don’t begin in new line but respect width , height , padding and margin

**Box property:**

Content-box : total width = width+padding+border



Border-box : total width = width(total width + padding+border) – padding applied within the element



**Visibility property :**

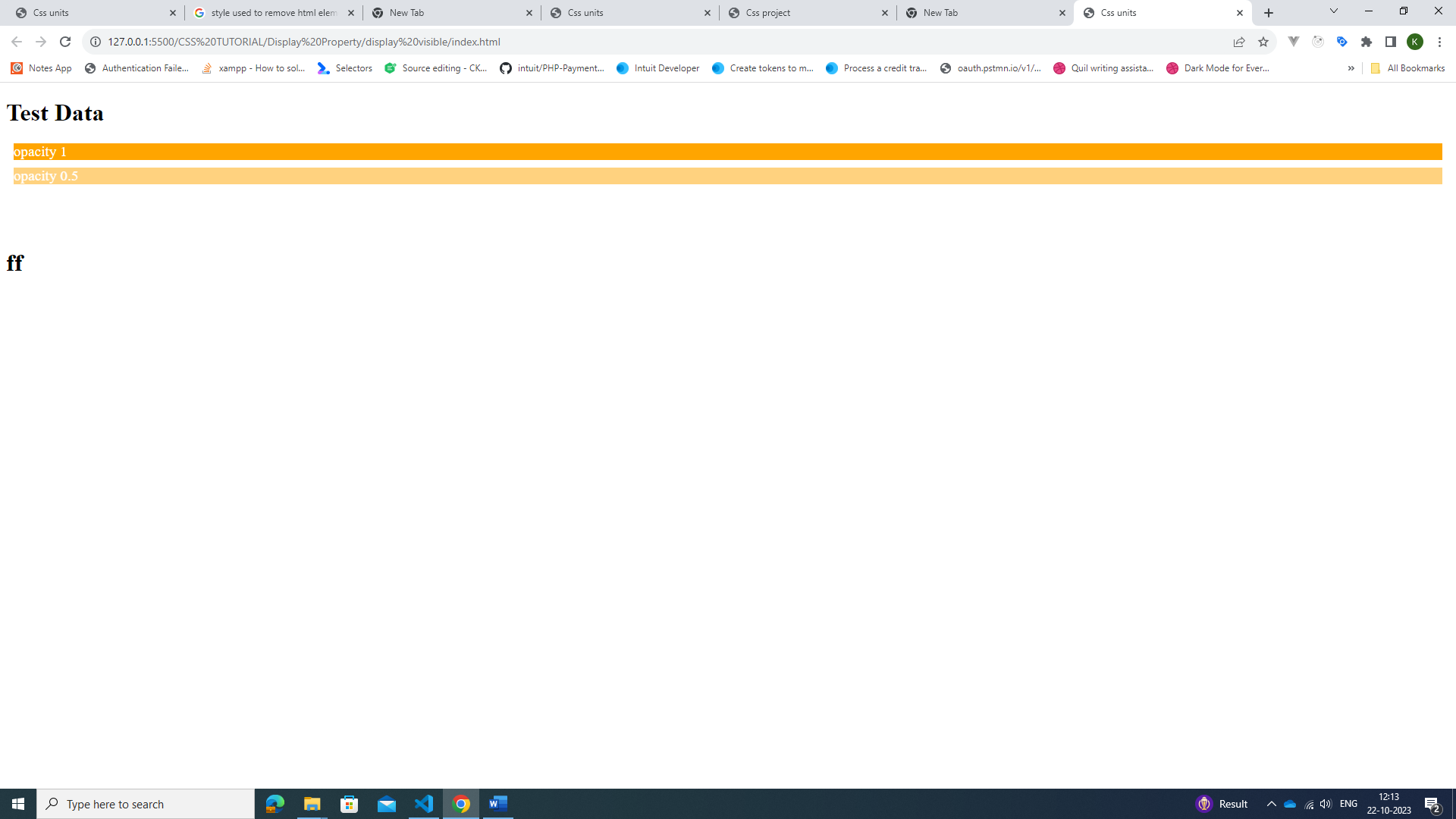
display : none – the entire content will be removed from page including space allocated for element . entire element will be removed from dom

visibility : hidden – the content will be hidden from page but space allocated for content will still remain in page

opacity : 0.5 – content will remain in page with blur

opacity : 1 - - clear content will display in page

opacity : 0 – content will be hidden from page but space remains



Background images:

Background : url();

   /\*  background-repeat: no-repeat;

    background-repeat: repeat-x;

    background-repeat: repeat-y;

    background-repeat: space; we create space b/t repeat images \*/

    background-repeat: no-repeat; /\* we don’t repeat the image ... this is recommended in most cases \*/

**repeat-x** – means it repeats in x axis

**repeat-y** - means it repeats in y axis

**no-repeat** – the image will not repeat

**space** – means it repeats and there will be space b/ images

**round** - the image will be repeated only when the second image have full width to occupy else the existing image will be stretched

**background -size**:

when we use no-repeat it is recommended to fix the size of image.

background -size : cover – this will cover the size of image for whole div(if image is small then image will be stretched)



background -size : contain – the image will be fixed to the size and if repeat is given then image will be repeat with its original fixed size



**background-position**

background-position:center – this will set the image to center

background-position : top – set image to top(bottom,left,right)

background-position : 10% 20% ; 10% to left and 20% bottom

**background-attachment:**

background-attachment:fixed - this will not scroll the image it will only scroll text associated with image

background-attachment:scoll - this will scroll the image with div when we scroll the div

**linear gradient :**

linear-gradient(to top, red,blue,green) – red will move from bottom to top

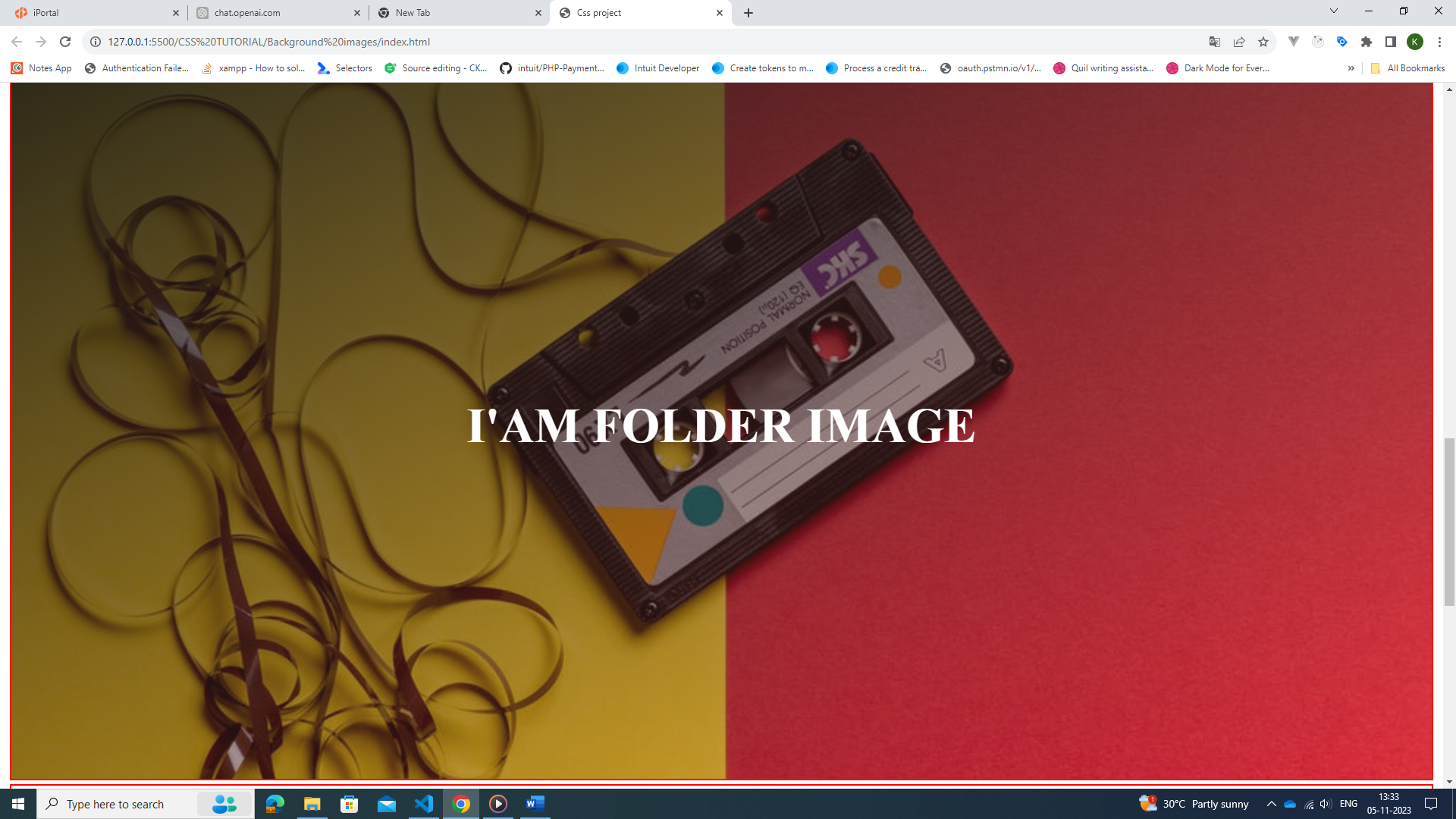
linear-gradient(to top left, red,blue,green) - move from bottom to top and right to left

linear-gradient(315deg, red,blue,green) – specify the degree of how color can be applied

linear-gradient(red 20%,blue 10%) – specifiy how much percentage each color should cover the div

linear-gradient(rgba(0,0,0,0.2),rgba(0,0,0,0.9)) – specify the opacity for color given

background image overlay :



Creating overlay to image with linear gradient combined with image

Background:linear-gradient(red,blue),url(‘’),

Background:linear-gradient(rgba(0,0,0,0.3), rgba(0,0,0,0.9)),url(‘’),

Shorthand property to combine all background property

Background:linear-gradient(rgba(0,0,0,0.3), rgba(0,0,0,0.9)),url(‘’) background-position(center,top,bottom,left)/background-size(cover/contain) background-attachment(fixed/ scroll) background-repeat(no-repeat,round,repeat-x,repeat-y,space),

Background:linear-gradient(rgba(0,0,0,0.3), rgba(0,0,0,0.9)),url(‘’) center/cover fixed no-repeat,

Online css linear gradient generate tool:

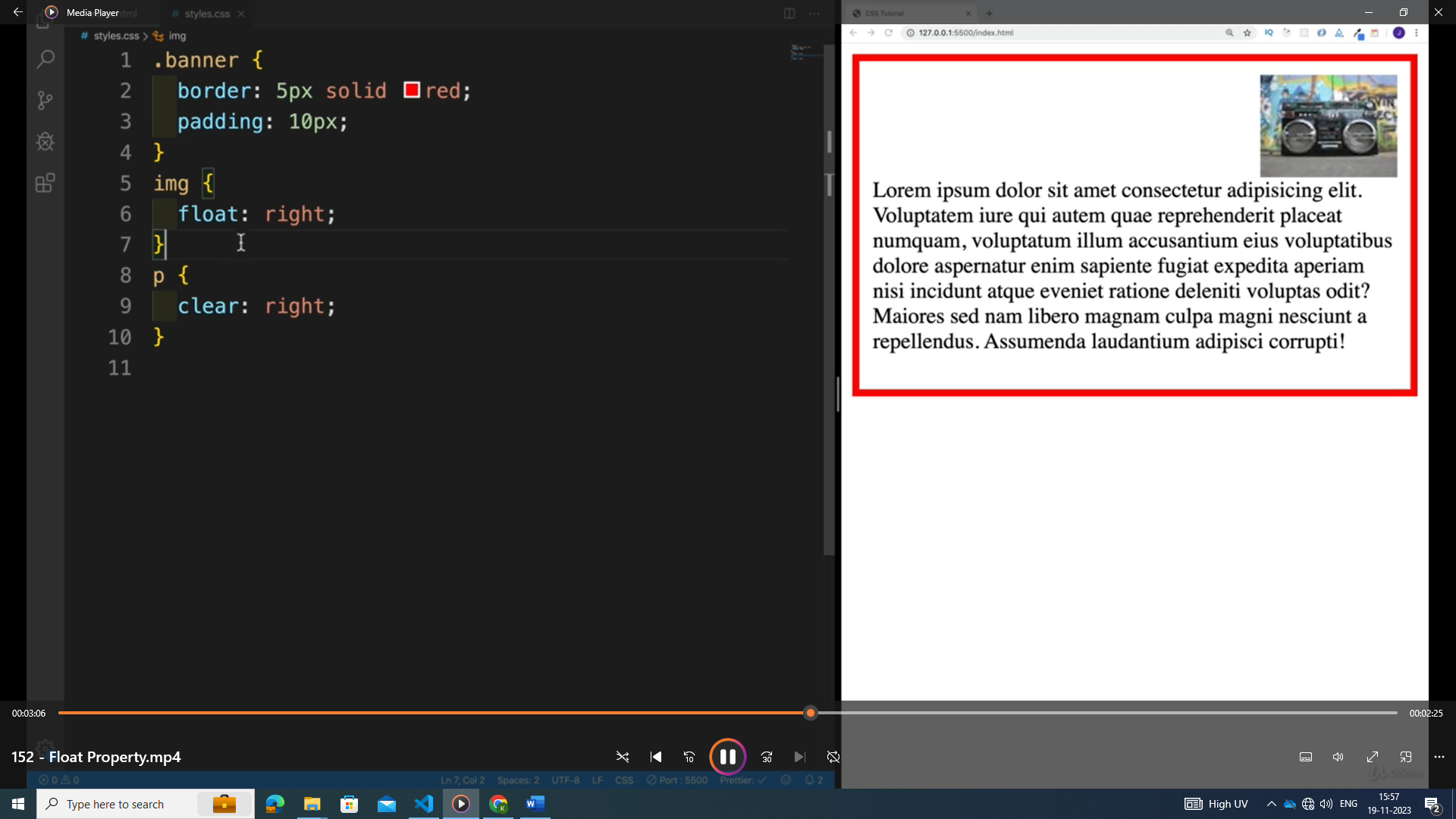
<https://www.colorzilla.com/gradient-editor/>

**float property:**

**float : left;**

**clear:left; --** this will start the next element in new line i.e element next to element with property left

values : left,right,both – both will start the next element always it is safer value to make next element appear In new line



use float to create multi column layout

3 divs with class name one,two,three

.One{

Width:30%;

Float:left;

Height:150px;

}

.two{

Width:30%;

Float:left;

Height:150px;

}

.three{

Width:30%;

Float:left;

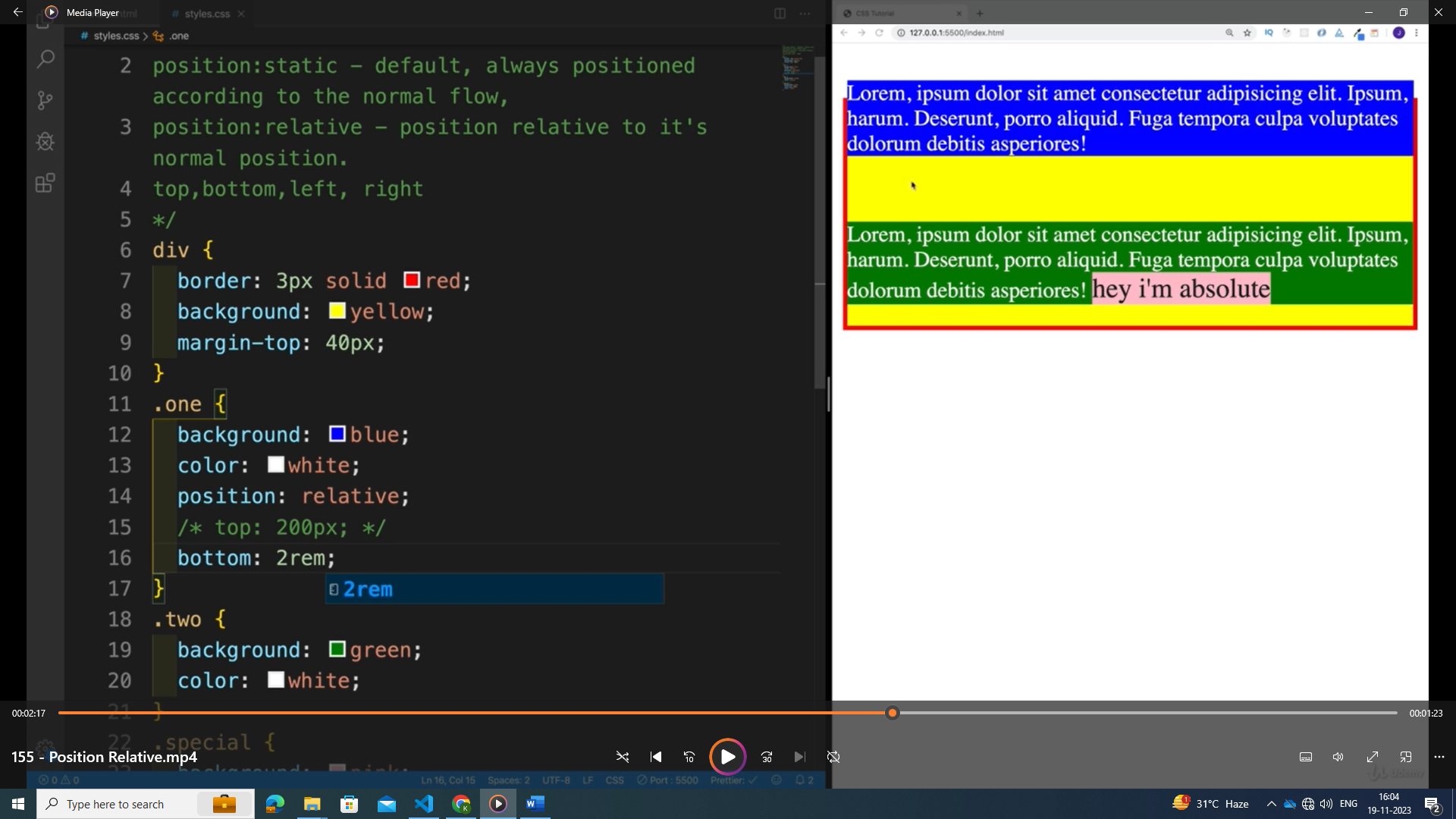
Height:150px;

}

**Position property:**

Position : static – the element will be displayed in browser in order which is defined in html

Position:relative – the element will be positioned relative to its normal position with top,bottom,left,right property i.e when we give top:200px Is that element will be 200px down from its normal position



Eg: .one class is moving 2rem up from normal position.

Note : difference between position relative & absolute :

When we move the element using relative the space **will not be filled by another element when we use position : relative**

When we use position : absolute then the left over space created by element with position : **absolute will be filled by element next to the element**. i.e other content will be adjusted to fill out the gap created by position : absolute

**position : absolute** – relative to the parent with position:relative so when we need to use **position : absolute then for sure its parent should have position : relative**

**Note : If non of the parent value don’t have position:relative then the main parent with position : relative is body**

Body is always position : relative

**Position : fixed : -** it is relative to viewport(Screen) stay as we are scrolling

It Is mostly used in nav bar

**Similar to position : absolute we are going to use other element to fill the space**

**Position : fixed**

**Top : 0 --** this means the element will be fixed to top of viewport i.e screen

**Position : sticky** – it will initially relative and once the position is met in viewport then it sticks

Top,left,bottom,right – top:0 is good value since if we meet the element then the element will be set to position:fixed

Eg: position : 20px;

distance b/t top of element and div element is 20px then the div element will be changed to position:fixed

**media queries :**

* **we should start the design from mobile view first**
* **min-width - start from**
* **max-width – upto that particular width**

**syntax :**

@media screen and (min-width:600px)

{

.banner{

}

}

Note : when particular property outside of media query is not overridden then those property will remain for any type of media query

**z-index:**

used to determine which element should be shown first in the screen

the element with high z-index will be displayed first in the screen

**Pseudo element:**

**::before,::after**

P::before{

Content : ‘text’;

Color:red;

Font-size:100px;

}

P::after{

Content : ‘’,

Width : 100vw;

Height :20px;

}

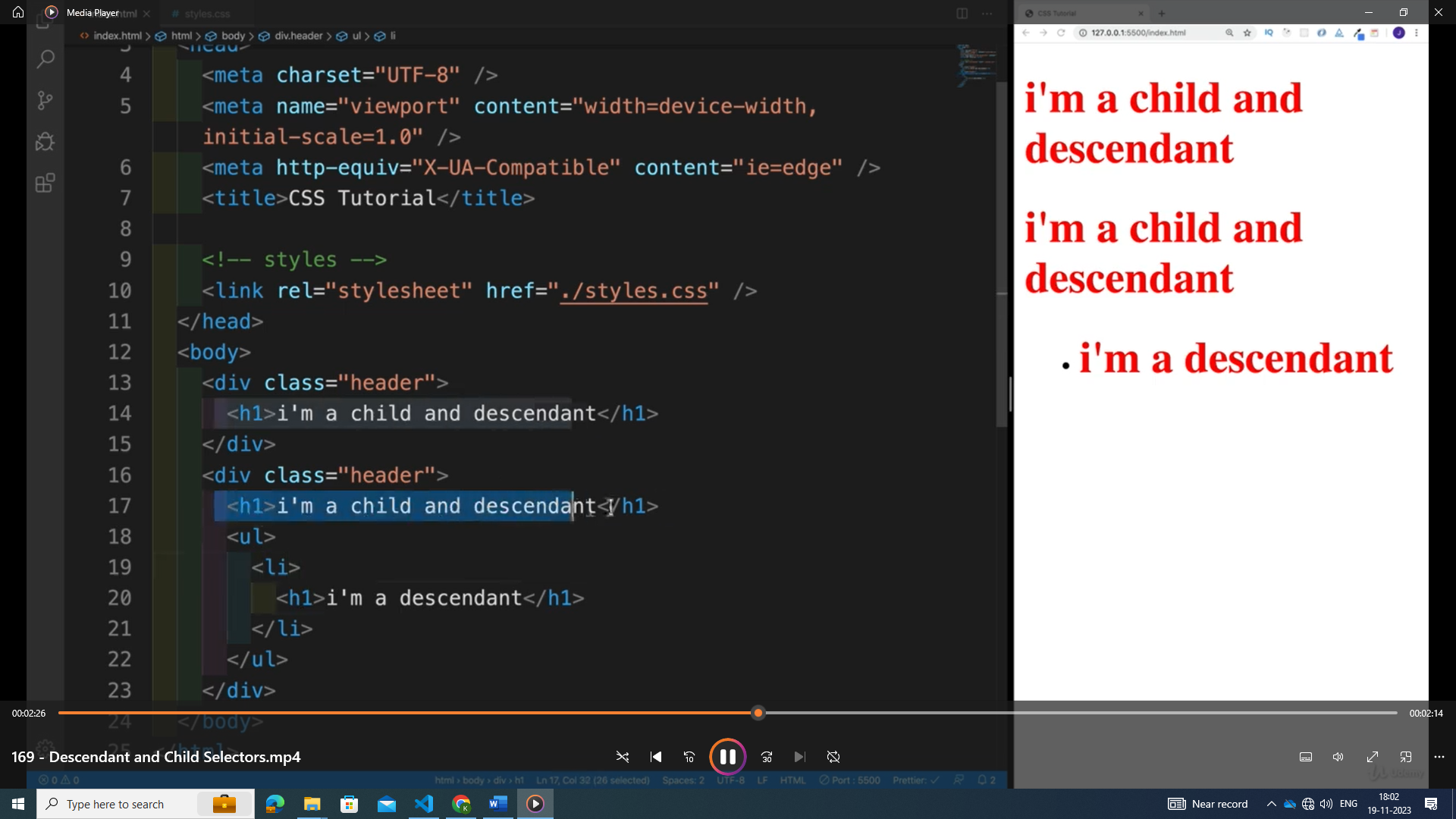
**Note : content property is mandatory for the before and after psudo elements**

**Inset property :**

Inset : top right bottom left

**Selectors:**

**Decedent selector:**



**Disdent selector :**

div h1{

}

--select all child and sub child of div

**Child selector :**

div>h1{} -selector only direct children

**first:**

::first-line and ::first-letter

::first-line – this is going to selector only first line

::first-letter – this will select only first letter

**Pseudo classes:** keyword added to selector that specify special state of the selected element

Some element have specific pseudo classes like links but it can be used in any element .

We can use pseudo classes on classes also like .header::hover

Eg:

When state of element changes then the action happens

**Links have pseudo classes:**

a:link{

}

Above is applied to link which is not visited yet

Note : <a href=”#”></a> - Is already visited

a:visited{

}

Above is applied when a link is visited

a:active{color:blue}

root pseudo :

Transition and animation

Flex : - multiple dimension layout with different features

Css grid : most recent way of setting up layout

* flex – it is used in some usecases to create multiple column layout
* css grid – much easy to use it
* we will be able to build multiple rows with 2 different dimensions – this is not possible with flex box
* i.e single div with both row and column

display : flex – block size element – it takes up whole size just like block element

display:inline-flex – it takes up only space that box have

display: grid

**column**

grid-template-columns : 30rem 20rem;

**row**

grid-template-rows: 30rem 20rem;

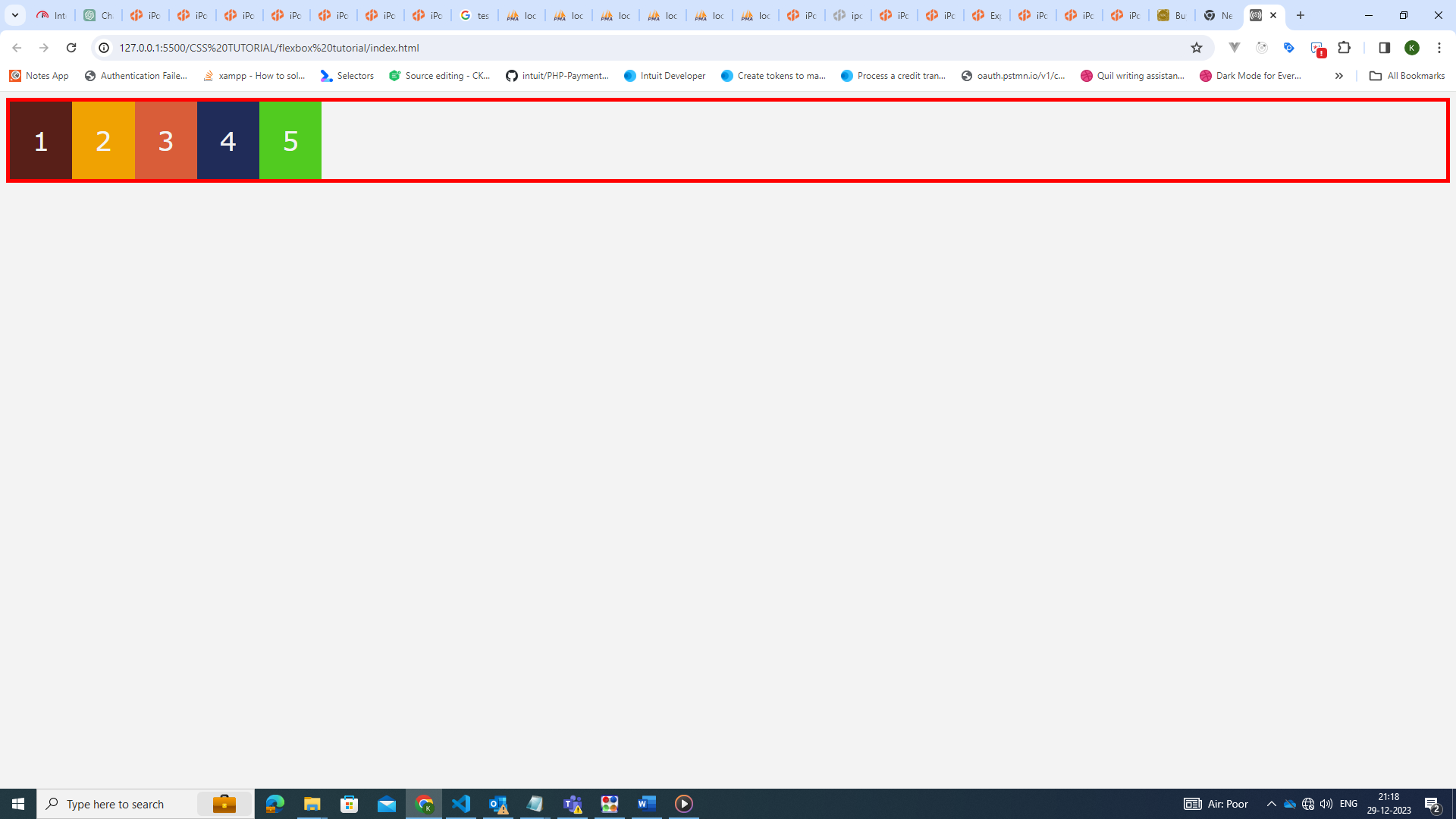
kinds of units that can be used with grid:

* px
* rem , em , % , auto ,fr
* auto – the size changes based on screen size

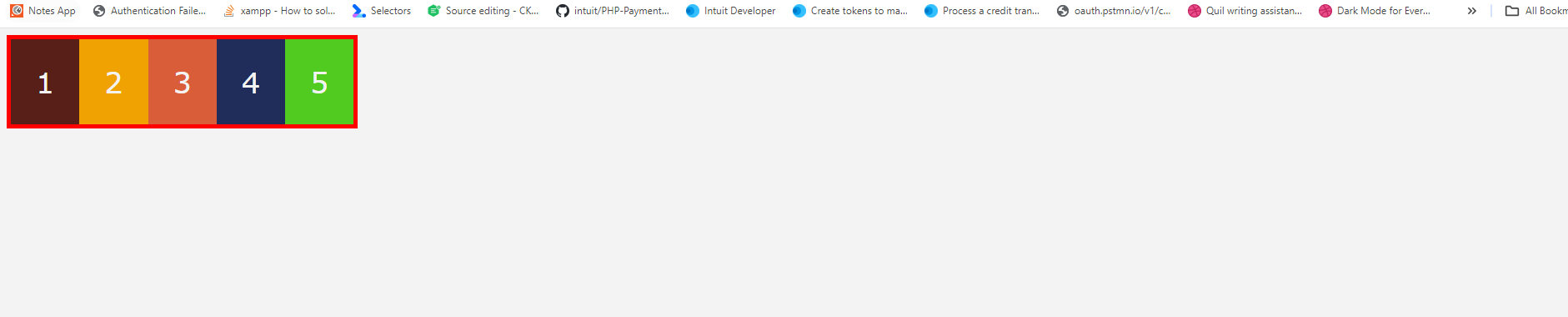
**Flex box:**

1. Flexible box model
2. Help spacing items in element
3. We can easily create navbars
4. Css grid replaced the flex box in many areas
5. All major browser supports the flex box

Display:flex



Display:inline-flex:



Properties in flex:

There are 2 types of properties

* The properties that will effect all child elements present inside parent container
* The properties that will effect each child element

Properties that effect the parent directly --always set on parent flex container i.e parent div for eg

* Flex-direction : row,row-reverst,column,column-reverse

**Best resources for font with css**

* Fontpair - https://www.fontpair.co/
* Pagecloud - https://typescale.com/
* Typescale – to get css of particular font

Colors:

<https://www.happyhues.co/>

<https://coolors.co/>

<https://noeldelgado.github.io/shadowlord/#f22ddd>

bootstrap

tailwind - <https://tailwindcss.com/docs/customizing-colors>

favion

https://favicon.io/favicon-generator/