CSS

What is css?

CSS stands for cascading style sheets. And mainly used to style the html elements

Why should we style?

To present the data inside the webpage beautifully,to organize the content for the end users to design layout properly

How do we apply css to elements?

There are three main ways to apply css to the elements such as

1. Inline css
2. Internal css
3. External css

Inline css:

Inline CSS is the technique to define the single element with the insert style sheets in an HTML document . It has the interactive and unique style to create a single HTML element; we can define the inline CSS on the style attribute.

Here is the basic syntax example :

<!DOCTYPE html>

<html>

<body>

<h1 style="color:blue;text-align:center;">This is a placeholder of heading</h1>

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

</body>

</html>

Example2 : <h2 style="color:red;margin-left:40px;">Inline CSS is applied on this heading. </h2>

Why is inline CSS used?

* An inline CSS is used to apply a unique style to a single HTML element. An inline CSS uses the style attribute of an HTML element.
* The major advantage of inline HTML is that it comes in handy and enables the multiple style attributes to an HTML tag with the defined CSS style of an element.

Disadvantages of Inline CSS:

* Adding inline CSS rules to each HTML element is time-consuming and gives messy results on the HTML page.
* Implementing multiple styling elements can affect the HTML page's size and download time.
* It provides different versions such as CSS, CSS 1, CSS 2, and CSS 3, which can be confusing for developers to make decisions about which version they need to refer to and can give the wrong result in the web browser.
* Fragmentation: It gives the browser an incompatibility issue, i.e., it may work with one browser and may not work on another type of browser.
* Lack of security.

<https://www.simplilearn.com/tutorials/css-tutorial/internal-css>

Internal CSS:

* Internal CSS is one of the most widely used CSS forms for changing, styling, and modifying the unique styles of a single web page. You can use the internal CSS by integrating the <style> element in the <head> section of a HTML web page.
* Internal CSS can be applied to the whole web page but not on multiple web pages and you can style several web pages by using the same code on every page.

How to Use Internal CSS?

* Internal CSS is a way of adding the CSS codes in the <style> element of <head> section of the document. All the changes done by the internal CSS can be applied only to a single web page. You can use internal CSS for one-page websites or projects where you do not want to use external CSS.
* Internal CSS is only beneficial for single-page HTML websites. If you use internal CSS in your HTML web page, you cannot reference this stylesheet to the <link> elements. It can make the modification and changes in a website more complex and challenging.

<https://www.javatpoint.com/internal-css>

* Internal CSS is not favorable for large projects where you are required to have more than one web page, therefore, the most common way to add CSS, is to keep the styles in external
* What is the difference between inline and internal and external CSS?
* Inline CSS is used to style a specific HTML element. Internal CSS is used to style a specific HTML page. External CSS is used to change the look of an entire website by changing just one file. You can write inline CSS using the style attribute.

What is the difference between inline CSS and internal CSS?

Inline CSS styles are included within the HTML document and are specific to individual HTML elements, allowing for targeted styling. Internal CSS styles are included within the head section of an HTML document and apply to the entire document, allowing for consistent styling across multiple elements.

SELECTORS

What is selector?

Which helps the developer to select an element.

why do we select?

To apply the properties to an element.

why should we apply ?

to present the data in a beautiful way.

we have different types of selectors in css. they are

* Simple selectors (select elements based on name, id, class)
* Combinator selectors (select elements based on a specific relationship between them)
* Pseudo-class selectors (select elements based on a certain state)
* Pseudo-elements selectors (select and style a part of an element)
* Attribute selectors (select elements based on an attribute or attribute value)

There are two main steps involved in css

1)select an element to apply the properties

2)adding property:values to style the elemnt

Types of selectors:

1) Global selector/universal selector:

* It can be represented as (\*) .It can select any element irrespective of their type.
* \* is a selector which represents every element ,whatever the property we are giving in the global selector it will applicable every where
* When we are going to use:
* Requirement demands to select every element in a web page we go for global selector

Syntax:

\*{

property :values;

}

Example:

\*{

color:red;

font family : arial;

}

Note:

1. We should keep html code seperately,CSS code seperately.
2. It's look clean,as long as application size increases and also easy to maintain.

2)Tag/Element selector:

Which helps us to select an element using their tag / element name

Syntax:

Tag name {

property :values;

}

Example:

Section {

border:1px solid green;

border-radius: 10px;

padding :20px;

}

3)Class selector:

* (.) represent class in css
* we have to give our own class name ,relevant class name have to give
* Class selector select an element with a specific class attribute
* Class can be duplicated
* Class can be reusable

When will we use class selector?

If we use same selector more than once within a page or a site (or) any property single or multiple properties adding multiple elements then we go for class selector

or

A group of properties planning to read again and again for multiple places to planning to use set of properties multiple times.

or

whenever you wanted to use properties to multiple elements we go for class selector

syntax:

.class-name{

proprerty: values;

}

Example:

.lop-fig {

border:1px solid red;

}

.pop-fig {

border:5px dotted yellow;

}

4) Id selector:

* # represents the Id selector
* It cannot be reusable
* It cannot be duplicated

Syntax:

# Id-name{

property : value ;

}

Example:

# lop {

background-color : orange;

}

# pop {

background-color : brown;

}

Difference between class and id selector ?

Note:

1. If any rule in css must be added to html element.
2. Id cannot be repeated.

5) Attribute selector:

* Based upon the attribute nature attributes are divided into 5 types

i) Has attribute:

Syntax:

[ attribute-name]{

property: value ;

}

ii) Attribute value equals to:

Syntax:

[ attribute-name =' ']{

property: value ;

}

Example:

iii) Attribute value start with:

Syntax:

[Attribute-name ^=' attribute-value-starts-with ']{

property: values;

}

iv) Attribute value ends with:

Syntax:

[Attribute-name$=' Attribute-value-ends-with ']{

property: values;

}

v) Attribute contains:

Syntax:

[Attribute-name\*=' Attribute-value ']{

property: values;

}

6)Relationship Selectors:

* Grand parent:

Immediate or direct childrens are fig1,fig2,fig3,fig4

ch1,ch2,ch3,ch4,ch5 are also a childrens of grand parent but not immediate childrens, these are called decendant childrens

* Siblings:

For fig1 ------>fig2,fig3,fig4 are siblings

* immediate siblings:

fig1-------fig2 --------adjacent sibling

fig2-------fig3--------adjacent sibling

fig3-------fig4--------adjacent sibling

fig4------- No immediate or adjacent siblings

* General siblings:

fig1----fig2,fig3,fig4

fig2----- fig3,fig4

fig3-----------fig4

fig 4---------No general siblings

Note:

In css it will not select the previous sibling ,only forward direction it will take

1) Parent and child relation

i) Direct childrens:

syntax:

selector>another-selector{

property:value;

}

Example:

.lop>figure{

border:10px doubled orange;

}

ii)Decendant childrens: In the scence it can be direct or indirect. it will cover both

Synatax:

Selector(space)another-selector{

Property:value;

}

Example:

.lop figure{

border : 10px dashed golden;

}

Note:

Direct &decendant both are having a same precidency. When bpth of them are in same precedency then again browser is going to consider them how they ordered ,According to order they will be placed.

2) Sibling relation

i) Adjacent sibling:

syntax:

selector+another-selector{

Property: value;

}

Example:

section+h2{

border: 5px dotted violet;

}

ii) General sibling

syntax:

selector~another-selector{

Property: value;

}

Example:

section~h2{

border: 4px dashed purple;

}

7) Psuedo selectors:

A pseudo-class is used to define a special state of an element.

i) psuedo class selector:

Syntax:

Selector:state-of-element

Example:

figure:first-child{

padding:30px;

background: linear-gradient(orange,white,green);

}

figure:last-child{

background: linear-gradient(orange,white,green);

}

figure:nth-child(odd){

background: linear-gradient(to left, orange,white,green);

}

For nth child ------we have to pass index of an element(index order)

nth child(even) or (2n)---------even number elemnts will be selected

nth child(odd) or(2n+1)----------odd numder elements will be selected

ii)psuedo element selector:

Syntax:

Selector::portion -of- element

Examples:

p::first-letter{

font-size: 40px;

}

p::before{

content: "\*\*\*\*";

}

p::after{

content: "\*\*\*";

}

8) Group Selectors:

The CSS grouping selector is used to select multiple elements and style them together. This reduces the code and extra effort to declare common styles for each element. To group selectors, each selector is separated by a space.

Syntax:

selector,another-selector,another-selector{

property:value;

}

Example:

p,h1,h2{

color:yellowgreen ;

}

Note1:

* In a live application design & functionality part are seperate
* Design:it is all about look&feel of aweb page
* Functionality:The moment when user start interacting to an element & the element is doing some sort of actions behind the scene
* By using HTML & CSS we can just allowed to design the element & cannot even add any functionality to it.
* Functionality can be added by using JAVASCRIPT

Note2:

* For a single element there are multiple selectors and common property value given, out of all the selectors in css ID selector takes highest precidency.
* PRECIDENCY:

If a single element being selected with multiple selctors with different properties,which property value should get affected to the element caled precidency.

What is the use of !important and give an example?

!imporatnt:

The !important rule in CSS is used to add more importance to a property/value than normal. In fact, if you use the !important rule, it will override ALL previous styling rules for that specific property on that element!

EXAMPLE:

p {

background-color: red !important;

}

BOX MODEL

Box model is a fundamental concept in css.That describes how elements are rendered on a webpage.It consists of four properties they are

1. Margins
2. Border
3. padding
4. content

Why do we take care about the rendering part of an application elements?

* To present the element in a properway
* To make an element inside a layout in a beautiful way

Typhography:

For an html some default properties are being applied by a browser,That concept we will call it as typography

Scenario: I dont want browser based properties to imapacted my element to remove these properties we can override the properties

NOTE:

The moment when you are giving '0' value to any property in css there is no need of specifying the unit of measurement

Shorthand properties:

Shorthand properties are CSS properties that let you set the values of multiple other CSS properties simultaneously.

Benfits of using shorthand properties:

* it will reduce the number of lines of code
* It will be easier for us to write a code

MARGIN:(margin):

It is a property and it can defines increases (or) decreases the space outside the element

* It can accept various units like pixels (px),percentage(%),ems (em) or even negative values
* sub properties of margin:

Margin-top

Margin-right

Margin-bottom

Margin-left

* If we give one value it will take four sides
* EX1: margin:0;
* Ex2: margin:10px 20px;(top &bottom,right&left)
* EX3: margin:10px 20px,30px; (top,right&left,bottom)
* Ex4: margin:10px 20px 30px 40px (top,right,bottom,left)
* EX5:margin:0 auto based upon the width of an element it will automatically adjust
* When we are increasing the margin outside of the element is increasing.

BORDER(border):

* To set limitation or border to an element
* It includes three sub properties
* Border-width: Specify the thickness of the border like (thick,thin,medium)or specify units(px,%,em)
* Border-style:Determines the appearance of the border like (dotted,dashed,doubled,solid)
* Style is most important , if you just give the border width it wont works
* Style and width are mandatory property values but color is not important
* Border-color: It sets the colour of the border
* \*\*\*what happen if you are not giving color( Whatever the text color it is there border is also applied\*\*\*
* Example: border:1px solid red;
* If we want to convert sharp edges to smooth edges we use border-radius
* Example: border-radius:10px;

PADDING:(padding):

* Increases the space within the element.It creates an inner space within the element
* Padding can accept all values in units like pixels
* It wont accept the negative numbers
* Padding also sub catagorized as

Padding-top

Padding-right

Padding-bottom

Padding-left

* Exmple: padding: 20px;
* Example2:padding : 10px 20px 30px 40px;

CONTENT:

* The content refers to the actual area occupaid by the elements text(or) other media(images,videos etc..)
* It is not directly controlled by a specific css property,but it represents the space where the elements content is rendered
* The size of the content is affected by the dimensions of the element,its padding,border and margin

Dimension properties

Dimension properties are useful in order to set the width&height of an element

Examples for width:

p{

border:1px solid red;

padding;20px;

width: 80%;

}

EX1:

p{

border:1px solid red;

padding: 10px;

width:80%;

min-width: 40%;

max-width: 50%;

height: 90px;

}

Example for height:

h2{

border: 3px dashed purple;

padding: 10px;

height: 60px;

min-height: 30px;

max-height: 70px;

width: 50%;

}

Responsive Web Design(RWD)

POSITION

* If you want move an elemnet across the page ,position property is the one to move across page
* When you are creating alayout please dont use position property
* If you create a layout using position ,Creating a webpage is not going to distrubed when you add some new elements then entire layout is going to be distrubed

Position properties are:

Static: If the element position is static then subproperties are position like top,right,bottom,left are going to work

Absolute:

* It converts block level element into inline
* if the element doesn't have parent always it listen to the browser view port
* If the absolute eleemnt has a parent it is not going to listen to browser view port it listen to the parent

Relative:

* It is not going to convert block level into inline
* If we want to move an element by using ralative it moves from the current position that to X-axis(or) y-Axis by using position subproperties

Fixed:

* It is going to listen to the screen
* It is convert block level into inline
* It is always fixed to the screen

Sticky:

* Elements are fixing to theheader portion of the particular page when you scroll downor up If we use subproperties

Z\_index:

* If we want to change the stack order of an element we use z-index
* Z-index it takes highest value in the first
* lowest value for the last

Display properties in css

* The display property specifies if/how an element is displayed.
* Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements:

* A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).
* The <div> element is a block-level element.
* Examples of block-level elements:
* <div> <h1> - <h6> <p> <form> <header> <footer> <section>

Inline Elements

* An inline element does not start on a new line and only takes up as much width as necessary.
* This is an inline <span> element inside a paragraph.
* Examples of inline elements:
* <span> <a> <img>

Display: none;

* commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.
* The <script> element uses display: none; as default.
* Override The Default Display Value
* As mentioned, every element has a default display value. However, you can override this.
* Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.
* A common example is making inline <li> elements for horizontal menus:

Example

li {

display: inline;

}

Display-flex

* The flex property is a shorthand property for:
* flex-grow
* flex-shrink
* flex-basis
* The flex property sets the flexible length on flexible items.
* Note: If the element is not a flexible item, the flex property has no effect.

flex-grow

* A number specifying how much the item will grow relative to the rest of the flexible items
* The flex-grow property specifies how much the item will grow relative to the rest of the flexible items inside the same container.
* div:nth-of-type(1) {

flex-grow: 1;

}

flex-shrink

* A number specifying how much the item will shrink relative to the rest of the flexible items
* The flex-shrink property specifies how the item will shrink relative to the rest of the flexible items inside the same container

div:nth-of-type(2) {

flex-shrink: 3;

}

flex-basis

* The length of the item. Legal values: "auto", "inherit", or a number followed by "%", "px", "em" or any other length unit
* The flex-basis property specifies the initial length of a flexible item.

div:nth-of-type(2) {

flex-basis: 100px;

}

Flex-wrap

* The flex-wrap property specifies whether the flexible items should wrap or not.
* nowrap --------Default value. Specifies that the flexible items will not wrap
* wrap --------Specifies that the flexible items will wrap if necessary
* wrap-reverse Specifies that the flexible items will wrap, if necessary, in reverse order

div {

display: flex;

flex-wrap: wrap;

}

Flex-flow

* The flex-flow property is a shorthand property for:
* flex-direction
* flex-wrap

Flex-direction:

* The flex-direction property specifies the direction of the flexible items.
* row Default value.--------- The flexible items are displayed horizontally, as a row
* row-reverse --------- Same as row, but in reverse order
* column ----------The flexible items are displayed vertically, as a column
* column-reverse --------- Same as column, but in reverse order

div {

display: flex;

flex-direction: row-reverse;

}

Display-grid

Transform

Transforms move or change the appearance of an element,

What is transform use?

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

It has 4 values they are

Transform-translate():

* If we want move an element across the page
* The translate() method moves an element from its current position

Transform-scale():

* whenever you want to zoom out or zoom in a particular element we use scale

or

* The scale() method increases or decreases the size of an element
* scaleX(): The scaleX() method increases or decreases the width of an element.
* scaleY():The scaleY() method increases or decreases the height of an element.

Transform-rotate():

* The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.
* It rotate the entire element

Transform-skew()

* The skew() method skews an element along the X and Y-axis by the given angles.
* The skewX() Method: The skewX() method skews an element along the X-axis by the given angle.
* The skewY() Method: The skewY() method skews an element along the Y-axis by the given angle.

Transition

Transitions make the element smoothly and gradually change from one state to another.

Why is transition important?

* Transitions help us move smoothly between activities in order to stay focused and engaged with each task.
* It is used to check the element how does it applying for the property old to new

How to Use CSS Transitions?

To create a transition effect, you must specify two things:

* The CSS property you want to add an effect to
* The duration of the effect

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

Transition:It is a short hand property of all transition

Transition-delay: The transition-dealy property specifies when the transition effect will start

Transition-duration:The transition -duration property specify how many seconds a transition effect takes to complete

Transition-property:It consist which property we have to transition

Transition-timing-function:

* The transition-timing-function property specifies the speed curve of the transition effect.
* The transition-timing-function property can have the following values:
* ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
* linear - specifies a transition effect with the same speed from start to end
* ease-in - specifies a transition effect with a slow start
* ease-out - specifies a transition effect with a slow end
* ease-in-out - specifies a transition effect with a slow start and end

Background-Properties

The CSS background properties are used to add background effects for elements.

Bakground properties are

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position
6. background (shorthand property)
7. background-color

background-color

* The background-color property specifies the background color of an element.
* If we set multiple has a background color dont call background-color
* Background is the property we use for multi color
* With CSS, a color is most often specified by:
* a valid color name - like "red"
* a HEX value - like "#ff0000"
* an RGB value - like "rgb(255,0,0)"

Ex:

background:linear gradient(orange,white,green)

background:linear gradient to right,(orange,white,green)

Opacity / Transparency

* The opacity property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent
* Note: When using the opacity property to add transparency to the background of an element, all of its child elements inherit the same transparency. This can make the text inside a fully transparent element hard to read

Transparency using RGBA

* If you do not want to apply opacity to child elements, like in our example above, use RGBA color values.
* 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).

div {

background: rgba(0, 128, 0, 0.3) /\* Green background with 30opacity \*/

}

CSS background-repeat

* By default, the background-image property repeats an image both horizontally and vertically.
* To repeat an image horizontally, set background-repeat: repeat-x;
* To repeat an image vertically, set background-repeat: repeat-y;
* The background-repeat property sets if/how a background image will be repeated.
* By default, a background-image is repeated both vertically and horizontally.

CSS background-repeat: no-repeat

* Showing the background image only once is also specified by the background-repeat property:

body {

background-image: url("img\_tree.png");

background-repeat: no-repeat;

}

NOTE: The background image is placed according to the background-position property. If no background-position is specified, the image is always placed at the element's top left corner.

CSS background-position

* The background-position property is used to specify the position of the background image.
* The background-position property sets the starting position of a background image.
* By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally
* background-position-------> Sets the starting position of a background image
* background-repeat---------> Sets how a background image will be repeated
* background-attachment---------> Sets whether a background image is fixed or scrolls with the rest of the page

background-attachment

* The background-attachment property sets whether a background image scrolls with the rest of the page, or is fixed.
* scroll---------> The background image will scroll with the page. This is default
* fixed--------> The background image will not scroll with the page

body {

background-image: url("img\_tree.gif");

background-repeat: no-repeat;

background-attachment: scroll;

}

Background-clip

* The background-clip property defines how far the background (color or image) should extend within an element.
* border-box----->Default value. The background extends behind the border
* padding-box The background extends to the inside edge of the border
* content-box The background extends to the edge of the content box

div {

border: 10px dotted black;

padding: 15px;

background: lightblue;

background-clip: padding-box;

}

background-origin

* The background-origin property specifies the origin position (the background positioning area) of a background image.
* This property has no effect if background-attachment is "fixed".

padding-box

* Default value. The background image starts from the upper left corner of the padding edge

border-box

* The background image starts from the upper left corner of the border

content-box

* The background image starts from the upper left corner of the content

cover

* Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges

contain

* Resize the background image to make sure the image is fully visible

Text properties

Text-align

* The text-align property specifies the horizontal alignment of text in an element.
* Left ---------- Aligns the text to the left
* Right --------- Aligns the text to the right
* center -------- Centers the text

Text-decoration

* The text-decoration property specifies the decoration added to text, and is a shorthand property for:

1. text-decoration-line (required)
2. text-decoration-color
3. text-decoration-style
4. text-decoration-thickness
5. text-decoration-line

text-decoration-line

* Sets the kind of text decoration to use (like underline, overline, line-through)
* text-decoration-line: overline;}
* div.b {text-decoration-line: underline;}
* div.c { text-decoration-line: line-through;}
* div.d { text-decoration-line: overline underline;}

text-decoration-color

* Sets the color of the text decoration
* The text-decoration-color property specifies the color of the text-decoration (underlines, overlines, linethroughs).

text-decoration-style

* Sets the style of the text decoration (like solid, wavy, dotted, dashed, double)
* The text-decoration-style property sets the style of the text decoration (like solid, wavy, dotted, dashed, double).
* solid -------------- Default value. The line will display as a single line
* double -------------- The line will display as a double line
* dotted --------------- The line will display as a dotted line
* dashed --------------- The line will display as a dashed line
* wavy --------------- The line will display as a wavy line

text-decoration-thickness

* Sets the thickness of the decoration line
* The text-decoration-thickness property specifies the thickness of the decoration line.

h3 {

text-decoration: underline;

text-decoration-thickness: 50%;

}

NOTE: Also look at the text-decoration property, which is a short-hand property for text-decoration-line, text-decoration-style, text-decoration-color, and text-decoration-thickness.

text-indent

* The text-indent property specifies the indentation of the first line in a text-block.
* Negative values are allowed. The first line will be indented to the left if the value is negative.
* length Defines a fixed indentation in px, pt, cm, em, etc. Default value is 0.

a {

text-indent: 50px;

}

div.b {

text-indent: -2em;

}

div.c {

text-indent: 30%;

}

Text-shadow

* The text-shadow property adds shadow to text.
* This property accepts a comma-separated list of shadows to be applied to the text.

h1 {

text-shadow: 2px 2px #ff0000;

}

Text-transform

* The text-transform property controls the capitalization of text.
* none ------------- No capitalization. The text renders as it is. This is default
* capitalize -------------- Transforms the first character of each word to uppercase
* uppercase ------------- Transforms all characters to uppercase
* lowercase -------------- Transforms all characters to lowercase

Word-spacing

* The word-spacing property increases or decreases the white space between words.

p {

word-spacing: 30px;

}

Word-wrap

* The word-wrap property allows long words to be able to be broken and wrap onto the next line.
* break-word Allows unbreakable words to be broken

div.b {

word-wrap: break-word;

}

## What is an Animation Timeline?

Historically an animation's timeline is the amount of time from when an animation starts to when it finishes, you may be used to seeing something like this.

.fade-element {

animation: fade-animation-name 300ms linear;

}

@keyframes fade-animation-name {

0% {

opacity: 0;

}

100% {

opacity: 1;

}

}

With this animation as soon as the animations start it takes 300ms for .fade-element to go from 0 opacity to 1 opacity. Because the animation function is linear, we know that if we look at this animation at 210ms opacity will be 0.7. Looking into different animation functions is beyond the scope of this blog, but let me know if you'd like me to cover it.

Imagine, then, if we could stop using time as our only timeline and instead could use percentage of an element that is on screen or how far we are between scroll position 400px and 600px.

This is exactly what an animation timeline is. We use two new CSS properties called

animation-timeline

animation-range.

.fade-element {

animation: fade-animation-name linear;

animation-timeline: scroll();

animation-range: 0 100vh;

}

## What are Scroll Animation Timelines?

As we sort of touched on in the previous section there are two types of scroll timeline they are

Scroll Progress Timeline

View Progress Timeline.

### Scroll Progress Timeline

This timeline is connected to the scroll position of a container, you can choose which container and also which axis.

To use the scroll timeline, you must set your animation-timeline to scroll(). As you can see, this is a CSS function and it takes two parameters.

First parameter:  
This is the element whose bar should be used.

* nearest (default) - first ancestor with a scroll in either plane.
* root - root element (body).

Second parameter:  
The axis to be used.

* block (default) - The block axis changes based on writing mode but in English this is vertical (up and down).
* inline - The inline axis again changes based on writing mode but in English is horizontal (left and right).
* vertical - Always vertical (up and down).
* horizontal - Always horizontal (left and right).

.fade-element {

animation: fade-animation-name linear;

animation-timeline: scroll(block root);

}

By default, a scroll timeline will use the entire page from 0% scrolled to 100% scrolled but that might not always be what you want. You might want to only run the animation for the first 50% of your site or the first 100vh. With the animation-range property it's easy to set this up.

.fade-element {

animation: fade-animation-name linear;

animation-timeline: scroll(block root);

animation-range: 20vh 80vh;

}

The animation-range property does change a little depending on which timeline type you use but with scroll it is simple to start scroll position and end scroll position with a default of 0% 100%.

### View Progress Timeline

A view timeline is based on a component and its relation to the view port. The timeline for this type is view() but only has one parameter which is the axis to be used.

.fade-element {

animation: fade-animation-name linear;

/\* this could be block | inline | vertical | horizontal \*/

animation-timeline: scroll(block);

}

As I alluded to earlier the animation-range property has changed here too as now we have 6 different ranges that can be used these are cover, entry, exit, entry-crossing, exit-crossing and contain. With these you can run different animations on an element depending on which range it is in.

cover  
This range starts when the element first touches the bottom of the screen and ends when the last part of the element exits the screen.

entry  
This range starts when the element first touches the bottom of the screen and ends when the last part of the element is fully on screen.

exit  
This range starts when the element first starts leaving the screen and ends when the last part of the element is fully off screen.

entry-crossing  
This is the same as entry.

exit-crossing  
This is the same as exit.

contain  
This range starts when the element is full on screen and ends when the element first starts leaving the screen.

[@bramus](https://dev.to/bramus) has a [great tool](https://scroll-driven-animations.style/tools/view-timeline/ranges/) you can use to help visualise what these mean.

.fade-element {

animation: fade-animation-name linear;

animation-timeline: scroll(block);

animation-range: entry;

}

The View() timeline introduces one last CSS property called view-timeline-inset. This property changes the location of the page's start and end points. It takes a percentage of the width or height, depending on the axis, and can be positive or negative.

.fade-element {

animation: fade-animation-name linear;

animation-timeline: scroll(block);

animation-range: entry;

view-timeline-inset: 10%;

}

## Demos

There was a little demo right at the start of the post just to whet your appetite, if you want to look over the code for that one here's the [sandbox](https://codesandbox.io/s/scroll-timeline-animation-falling-nd3syq) but here are another couple of demos to show what is possible.

### Scroll animation

In this demo I have a sticky header that is 100vh and I scroll down an animations plays revealing the content behind it. I also have to move the main content as we scroll in order to make sure it appears static.