DISPLAY PROPERTY

This property is used to define how an element should display. 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.

Inline -: When we set this value, the element does not start on a new line and only takes up as much width as necessary (we can not width/height it won't work).

Block-: When we set this value element always starts on a new line and takes up the full width available (we can set width/height).

Inline-block -: It is combination of inline and block value.It doesn't start on new line but we can set width and height.

WHAT IS FLEXBOX IN CSS

Flexbox is used for better one dimensional layout (row or column) in CSS. Flexbox well implements space distribution of items in the same direction. It arranges the alignment of items in the box through flexbox. This is a powerful way to create CSS layout in web page designing.

Features of Flexbox :-

- It is easy to create flexible layout in Flexbox. In this, it is very easy to arrange the items inside the container i.e. in left-to right, right to left, top to bottom or bottom to top direction.
- The space between items can be easily handled.
- The alignment and order of items can also be maintained properly.

How does the Flexbox layout module work :-

There are 2 entities in flexbox and they are:

1. Flex container :-

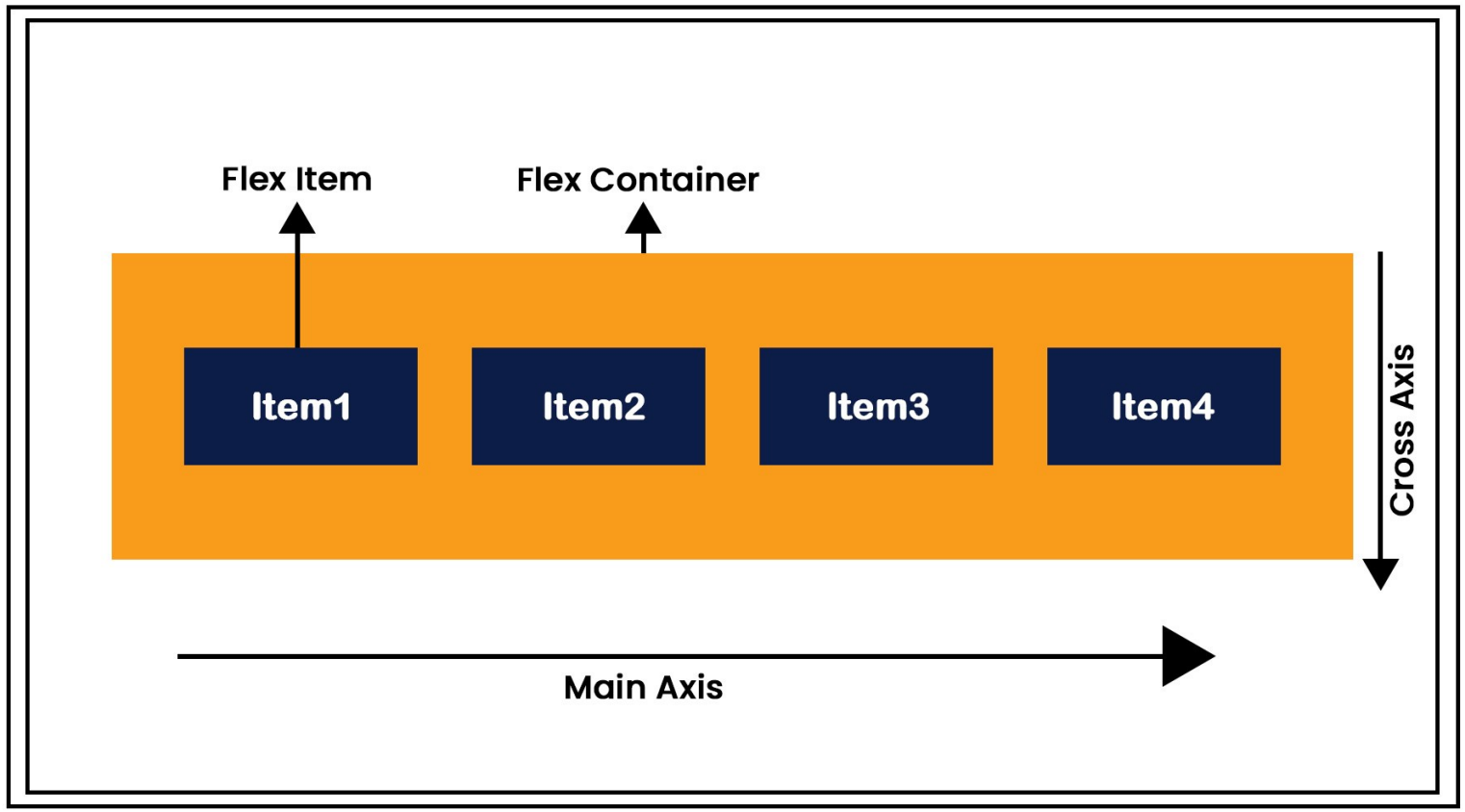
The parent container in which flex property will be placed is called flex container. In simple terms, it is a parent element.

2. Flex items :

The elements coming inside it are called flex items.

The second thing that comes in flexbox is its axes and that is:

1. main axis :- The direction of the main axis is from left to right.
2. cross axis :- The direction of the cross axis is from top to bottom.



Note :-To make a container flexbox, flex value is given in the display property.

Flexbox has some properties which are defined for flex container some properties which are defined for flex items.

Let us now understand all those properties.

FLEX CONTAINER PROPERTIES :-

1. Display
2. Flex-direction
3. Flex-wrap
4. Flex-flow
5. Justify-content
6. Align-items
7. Align-content

DISPLAY :- This is the first property. Any property of flexbox will not work without placing it in the parent container box. This indicates that all the items in this box will be in flex style.

This happens in two ways :-

Flex :- The element in which we use the display:flex property works like the element display:block property.while its content (child element) maintains its flexbox properties

Inline-flex:- The element in which we use the display:inline-flex property works like the element display:inline property.while its content (child element) maintains its flexbox properties

FLEX DIRECTION :-

Through flex direction property we set the direction of flex items. That is, how will the flex items be arranged in the flex container.

It has Four values :-

1. **Row:-** In this all the flex items are arranged in a row.
2. **Row-reverse:** In this, all the flex items are arranged in a row but its order starts from the reverse direction. That means all the items are arranged from right to left.
3. **Column:-** In this all the flex items are arranged in one column.
4. **Column-reverse:-** In this, all the flex items are arranged in one column but its order starts from the reverse direction. That means all the items are arranged from bottom to top.

Note: The flex-direction property is written with display:flex.

Flex wrap property :-

When flex items are more then it starts overflowing from the flex container. To overcome this problem, flex-wrap property is used.

The flex-wrap property has Three values and they are :-

1. no-wrap: This is the default value of flex-wrap. Even if there are more flex items at this value, it does not wrap them.
2. wrap: If flex items are overflowing on this value then it wraps it.
3. wrap-reverse: This works opposite to the wrap value. This wraps the overflowing flex items from the opposite direction.

Flex flow property :-

flax flow property is a shorthand property. It takes the values of both flex-direction and flex-wrap properties together.

Syntax :-

flex-flow: flex-direction flex-wrap;
ex: - row wrap ;

Justify-content property :- The justify-content property is used to align the flex items. Along with this, this property also helps in distributing the extra space inside the container among the items.

Its values are:

1. flex-start
2. flex-end
3. center
4. space-around
5. space-between
6. space-evenly

1. Flex-start :-
With this value, flex items remain aligned in the exact left direction in the container. This is the default value of justify-content.

2. Flex-end :-
With this value, flex items remain aligned in the right direction in the container.

3. Center :-

With this value, flex items remain center aligned in the container.

4. Space-between :-

If we want to divide the container space equally among flex items, then we can set the value of the justify-content property to space-between.

5. Space-around :-

If we want to divide the container space equally between flex items and also want space at the beginning of the first item and at the end of the last item, then we can set the value of the justify-content property to space-around. With this value, the space at the beginning of the first item and the end of the last item is half the space of the space between the remaining items. That is, if the space between the items is divided into 20px, then the starting space of the first item and the last space of the last item will be 10px.

1. **Space-evenly :-**

With space-even value, the space of the container is distributed equally among all the items. With this value, the starting space of the first item and the last space of the last item are also equal.

ALIGN ITEMS PROPERTY :-

The align-items property is used to align the flex items. Vertical alignment is given to flex-items through the align-items property. Through this property, flex items are aligned to the cross axis.

Its values are: -

1. Stretch 2. Flex-start 3. Flex-end 4. Center 5. Baseline

Stretch :- In this value, all the flex items inside the container are stretched to the entire height of the container. stretch is the default value of the align-items property.

Note :- To use this property, it has to be seen whether both the height and width of the flex items have been defined, if so, then it will not stretch, whatever is not defined, only that will stretch.

Flex-start :- With this value, all the items are aligned to the top position (cross-axis start) of the container.

Flex-end :- With this value, all the items are aligned to the bottom position(cross-axis end) of the container.

Center :- With this value, all items are aligned to the cross-axis center of the container.

Baseline :- With this value, flex items inside the container are aligned to the baseline of its content.

ALIGN-CONTENT PROPERTY :-

To align items property, a single row is to be used. While to align content property, a double row is to be used.
If your content is not coming in one line then it will come in another line then the space alignment of both the lines will be done by align-content property.

The values of the align-content property are :-

Flex-start :- With this value, all items are aligned to the top position (cross-axis start) of the container.
Flex-end -: With this value, all the items are aligned to the bottom position(cross-axis end) of the container.
Center -: With this value, all items are aligned to the cross-axis center of the container.

Stretch -: In this value, all the flex items inside the container are stretched to the entire height of the container. stretch is the default value of the align-content property.

Space-between -: If we want to distribute the extra space of the container equally among the flex items, then we can set the value of the align-content property to space-between.

Space-around-: If we want to divide the container space equally among the flex items and also want space at the beginning of the first item and at the end of the last item, then we set the value of the align-content property to space-around. Can do.

Space-evenly -: space-evenly value distributes the space of the container equally among all the items. Along with this, the starting space of the first item and the last space of the last item are also equal.

Note:- The align-content property works on multi-line. If flex items are on a single line then align-content starts working like align-items property. But its Three values space-between, space-around, space-evenly do not work on a single line.

Flex-items properties :-

1. **Order.**
2. **Flex-grow.**
3. **Flex-shrink**
4. **Flex-basis .**
5. **Flex.**
6. **Align-self .**

Order property :- The order of items is controlled through this property. It takes value integer. The default value is 0.

The following values are accepted by the order property :-

- A positive integer:

  The item is displayed after items with lower order values.(lowest order value will be visible first)

- A negative integer:

  The item is displayed before items with lower order values. (lowest order value will be visible last).

- 0:

  The item is displayed in its default order.

Flex-grow Property :-

1. The flex-grow property tells flex items how much space they can take up in the flex container as per requirement.
2. By default flex items take up as much space as the content inside it. The flex-grow property takes an integer value. The default value of all flex items is 0(zero).

Flex-shrink Property :-

3. The flex-shrink property allows flex items to shrink. The default value of this property is 1.

2. If we don't want flex items to shrink, we can set tab flex-shrink:0.

3. If we want to shrink a flex item more than other items, then we will have to increase its flex-shrink value.

Flex-basis Property :-

1. The initial size of flex items can be set through the flex-basis property. Its value can be set by pixel, percentage, rem or auto.
2. The default value of this property is auto. This value means that flex items will take up space according to their content.

Flex property :-

flex property is shorthand for flex-grow, flex-shrink and flex-basis.

Syntax -: flex: flex-grow flex-shrink flex-basis.

Default Value -: flex: 0 1 auto;

ALIGN SELF PROPERTY :-

1. The align-self property is used to control the alignment of individual flex items (We can align any individual element).
2. The align-self property specifies the alignment for the selected item inside the flexible container.
3. The align-self property overrides the default alignment set by the container's align-items property.

VALUES OF THIS PROPERTY ARE:

1. **Auto:** The behavior of the auto property will be the same as the property attached to the parent div. If the parent div has the align-items property centered. So auto property will also be the center.
2. **Stretch:** The element is stretched to fit the container.
3. **Flex-start:** The element is aligned to the top of the container.
4. **Flex-end:** The element is aligned to the bottom of the container.
5. **Center:** The element is aligned at the vertical center of the container.
6. **Baseline:** The element is aligned at the baseline of the container.

NESTED FLEX CONTAINERS WITH FLEXBOX

You can create two dimensional layouts by nesting a flex container inside another one. Flexbox is inherently a one dimensional layout model. Flex items within a flex container can be laid out either horizontally or vertically, but not both..

FLEXBOX WITH MARGIN AUTO

Using auto margins with Flexbox is an effective way to get all of the flexibility of CSS floats. It can be applied horizontally or vertically and it gives you more control of flex items along the main axis.

CSS FLOAT PROPERTY

When an element is rendered in the browser, the tab is positioned, by default, in the top-left direction. Many times there is a need that we want to set the elements in the right direction.
Like you want to keep the content on the left side and want to show the related image on the right side. In such a situation, float property comes in handy.

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

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

In its simplest use, the float property can be used to wrap text around images.

THE 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 can have one of the following values:

1. **None -** The element is not pushed below left or right floated elements. This is default
2. **Left -** The element is pushed below left floated elements
3. **Right -** The element is pushed below right floated elements
4. **Both -** The element is pushed below both left and right floated elements
5. **Inherit -** The element inherits the clear value from its parent

CSS Selectors

CSS SELECTORS :-

CSS Selectors are used to select an element in a webpage and give it style. There are many types of elements like paragraph, heading, input element in a webpage.

Selectors are used in the program to select the element as per requirement and to give style to it.

There are 5 types of CSS Selectors and they are:-

1) Simple Selectors.
2) Combinators Selectors.
3) Attribute Selectors.
4) Pseudo-Class Selectors.
5) Pseudo-Element Selectors.

SIMPLE SELECTORS

1) Simple Selectors :- Simple Selectors are those selectors which select the html element through its id, class, name and then give it style.

There are 5 types of Simple Selectors :-

A. Type or element Selectors (selects an element by element name or tag name).
B. class selector (selects element from CSS class).
C. id selector (selects an element by its id).
D. universal selector(*).
E. grouping selector.

A. Type or element Selectors :-

This is a special HTML element. It is also called Type Selector. In this you declare CSS Rule by making HTML Element a selector.

Example :-

```
p {
    color:orange;
 }
<p>
      Lorem ipsum, dolor sit amet consectetur adipisicing elit. Voluptatum,illo?
<p>
```

B. Class selector :-

Elements to which you want to apply a style rule. Class Define is explained by Class Attribute in all elements. And the CSS rule is written by putting full stop symbol (.) before the name of this class.

You can define a class to contain more than one element. All you have to do is define a separate class for each element with a name. Keep one thing in mind that you cannot start the class name with a number.

```
.test{
        color:orange;
      }
<div class="test">
    Lorem ipsum dolor sit, amet consectetur adipisicing elit. Ullam eius sit do
 </div>
```

A. ID SELECTOR :-

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example :-

```
#Test1 {
  text-align: center;
  color: red;
}
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
```

D. Universal selectors :-

Universal selectors are used when you want to apply the same style rule to all the elements available in an HTML document. Universal Selector is represented by * (Asterisk).

EXAMPLE :

syntax: * { property : value;}

```
* {
   color:orange;
  }
```

With this style rule, the color of all the elements available in a document will become orange.

E. Grouping selector :-

If you want to give the same style to many HTML elements, then grouping selector is used. And all the tags which are to be styled are written in one line, separated by comma. After that CSS rules are written. So all those styles will be applied to all those tags simultaneously.

Example :-

```
h1, h2, p {
        color:red;
        font-family:Arial;
                border:2px;
        }
```

Note :- In the example given above, CSS styling has been done together for h1, h2 and <p> tag.

COMBINATOR SELECTORS

2. Combinator Selectors :-

We can target elements through simple selectors, but there are some special elements which are difficult to target directly. Like some elements are inside other elements. Which may be difficult to select from class or id.

In such situations, combinator selectors come in handy. As the name suggests, combinator selector creates a relation between two selectors. The combinator selector is written between two selectors through a symbol.

There are 4 types of combinators selectors and they are indicated by their symbols.

A. descendant selector(space). B. child selector(>).
C. adjacent sibling selector(+). D. general sibling selector(~).

A. Descendant selector(separated by space) :-

- With Descendant Selectors, all its child tags are mapped with the tag, to give style. To give a similar style to all the child tags, descendant selectors are used. (With descendant selectors you can give the same style to all child tags simultaneously).
- In descendant selectors, two selectors are written by giving space symbol.
- Whether it is an immediate child or an indirect child, the CSS style will be applied to all child tags.

Example :-

```
div p {
        color:red;
        font-family:Arial;
            }
```

B. Child Selector(>) :-

Child Selector is used to style only those tags which are direct children of the Parent Element. Two selectors are separated by the > (greater than) symbol. Child Selector is used to match all the elements which are children of a specified element. It gives the relation between two elements. The element > element selector selects those elements which are the children of the specific parent. The operand on the left side of > is the parent and the operand on the right is the children element.

Syntax:

```
element > element
{
  // CSS Property
}
```

C. Adjacent sibling selector(+) :-

Adjacent sibling selector is used to style the element immediately following an element. In this, both the selectors are written by connecting them with +(plus) symbol. Meaning, the tag immediately following a tag will be called adjacent sibling.

Syntax:

```
element + element
{
  // CSS Property
}
```

D. General sibling selector(~) :-

All elements of the same type after an element will be called general sibling. Two general sibling selectors are added to the (tild) symbol.

Attribute Selectors :-

We can also target any element with attributes.

With attribute selectors we can select HTML elements or tags and through their attribute | Using only the attribute name or the value along with the attribute name can target the element and apply attractive styling.

The attribute selector is written through [ ] square bracket.

```
Syntax: Selector[attribute expression]

{

css style declaration;

}
```

Selector :-

can be anything, it can be any tag, id, class. Even if we do not take a selector, the attribute that will be targeted will be applied to all the elements.

Attribute expression :-

You can enter attribute name or any value along with attribute name in the attribute expression. The value of the attribute can also be given with some special conditions. Which we will understand with examples in attribute selectors types.

Attribute Selectors Types :-

Attribute or attribute's value, both can be used according to the selector. That is why Attribute Selectors are divided into 7 types.

1. **A[attr]** 2. **A[attr=val]** 3. **A[attr^=val]** 4. **A[attr|=val]** 5. **A[attr$=val]**
6. **A[attr*=val]** 7. **A[attr~=val]**

Note :- A represents an element or tag. attr means indicating the attribute and val means the value of that attribute.

Example: input[type] ->in this input is a tag. And type is the attribute of <input> tag.

1. A[attr] :- In this type, a special attribute of a tag or element is taken according to the selector. The [attribute] selector is used to select elements with a specified attribute.

2. A[attr=val] :- In this the attribute and its value is taken according to the selector.

3 A[attr^=val]:- Starting with Attribute and Special Values. ^ Symbol represents the beginning of a word. In this type of selector, the beginning of the attribute and its value is taken.

```
<style>
a[name^='it']
{
color:darkgreen;
font-weight: bold;
}
</style>
<a href="abc.com" name="it site" target="_blank">ABC</a>
<br><br>
<a name="used site" href="xyz.com">XYZ</a>
```

4. A[attr|=val] :- An element which has an attribute whose value is also given with hyphen and if we want to select it then we can do it using this selector.

5. A[attr$=val] :- In this type of selector, the last word of the attribute and its value is taken. Last word of Attribute and its special value. The $ symbol represents the last .

```
<style>
img[name$='flower']
{
border:5px solid blue;
}
</style>
<img src="lily.jpg" name="lili flower" />
<br><br>
<img src="rose.jpg" name="rose flower beautiful"/>
```

Note:- See in the example, the value of two imgs contains the word ‘flower’. But $ symbol takes the last word. That's why if there is word 'flower' at the end of the first img tag value then it will have style.

6. A[attr*=val] :- Wherever the value is in the attribute, it is taken according to the selector. That particular value is represented by * symbol.

```
<style>
img[name$='flower']
{
border:5px solid blue;
}
</style>
<img src="lily.jpg" name="lili flower" />
<br><br>
<img src="rose.jpg" name="rose flower beautiful"/>
```

7. A[attr~=val] :- In this type of selector, the value of the attribute taken must have a space before and after it.

```
<style>
img[name~='flower']
{
border:5px solid blue;
}
</style>
<img src="lily.jpg" name="lili flower"/>
<img src="rose.jpg" name="roseflower beautiful" />
```

PSEUDO CLASS IN CSS

Introduction to CSS Pseudo Classes :-

CSS pseudo classes are used to design special states of elements.

But sometimes there are some conditions and states about which there is no information in the document tree. Pseudo classes are used to target these conditions and states.

For example, you want to change the color of a paragraph to red when the mouse is hovered over. But you cannot do this with any normal CSS selector because this is a special state which is generated by the user and there is no information related to it in the document tree.

Pseudo Class Selectors List

:first-child	:focus	:link
:last-child	:checked	:hover
:nth-child()	:disabled	:visited
:nth-last-child()	:enabled	:active
:nth-last-of-type()	:required	:target
:nth-of-type()	:optional	
:only-child	:in-range	:lang()
:only-of-type	out-of-range	:root
:first-of-type	:read-only	
:last-of-type	:read-write	
:empty	:valid	
:not()	:invalid	
	:default	

Note :- Pseudo classes always start with colon (:)

Syntax of CSS Pseudo Classes :-

```
selector:pseudo-class
{
    property-name:value;
}
```

***Now let us understand all the selectors one by one ***

:First-child :- It selects those elements which are the first child of the parent.

In this pseudo class, the element which you are using as selector should be the first child of the parent element.

If the define element comes second position then this selector can not select that element. Meaning if another type of element comes before the define element then it will not work.

:First-of-type :- first-of-type pseudo class, you can select that tag which is written for the first time anywhere in the HTML document, or which is written for the first time inside any parent element.

Meaning, if any other type of element comes before the defined element, it will still work.

:Last-child :- If the last tag inside a parent element is to be selected, then the :last-child pseudo class is used.

In this pseudo class, the element which you are using as selector should be the last child of the parent element. If the define element comes second position then this selector can not select that element. Meaning if another type of element comes before the define element then it will not work.

:Last-of-type :- last-of-type pseudo class, you can select the tag which is written last anywhere in the HTML document, or which is written last inside any parent element.

:Last-of-type pseudo class works the opposite of :first-of-type pseudo class.

Meaning, if any other type of element comes before the defined element, it will still work.

nth-child( ) :- To select a particular child of a parent element, the :nth-child pseudo class is used. For example, there are 5 paragraphs inside a div tag and you want to style the 3rd paragraph, then you can do it through nth-child(3).

Any child tag can be targeted with this pseudo class. For example, if you want all odd numbered or even numbered children inside the parent tag, then you can add nth-child pseudo class.

We can also do it like this :-

:nth-child(2n) :-

Here 2n is :-

2 * 0 = 0

2 * 1 = 2 (second child element)

2 * 2 = 4 (fourth child element)

:nth-child(3n) :-

Here 3n is :-

3 *0 = 0

3* 1 = 3 (Third child element)

3* 2 = 6 (Sixth child element)

:nth-child(2n + 1) :-

Here 2n is :-

2*0 + 1 = 1

2* 1 +1 = 3 (second child element)

2* 2 + 1= 5 (fourth child element)

:nth-last-child( ) :- The :nth-last-child() pseudo class works the opposite of :nth-child(). It selects the child tag of the parent element by counting from the bottom. Meaning, the last tag of the parent element is the first tag for it and the last tag is the first tag for it.

We can also do it like this :-
:nth-last-child(2n) :-
Here 2n is :-
2*0 = 0
2* 1 = 2 (second child element)
2* 2 = 4 (fourth child element)

:nth-last-child(3n) :-
Here 3n is :-
3 *0 = 0
3* 1 = 3 (Third child element)
3* 2 = 6 (Sixth child element)

:nth-child(2n + 1) :-
Here 2n is :-
2*0 + 1 = 1
2* 1 +1 = 3 (second child element)
2* 2 + 1= 5 (fourth child element)

:nth-of-type( ) :- nth-of-type() pseudo class, we can select a tag with a particular number whether it is inside the parent element or in the HTML document.

Meaning, if any other type of element comes before the defined element, it will still work.

:nth-last-of-type( ) :-

nth-last-of-type() pseudo class, we can select a tag with a particular number whether it is inside the parent element or in the HTML document. But it works in the opposite direction, meaning it selects in descending order.

The : nth-last-of-type() pseudo class works the opposite of :nth-of-type( ).

Meaning, if any other type of element comes before the defined element, it will still work.

:only-child :- only-child pseudo class, you can select that tag which is written only once in a parent element.

:only-of-type :- only-of-type pseudo class, you can select that tag which is written only once in a parent element or in an HTML document.

Meaning, if any other type of element comes before the defined element, it will still work.

:empty :- If any html element is empty in html document. If there is no content and no space inside that HTML tag, then :empty pseudo class is used to select such element.

:not( ) :- If CSS is to be applied on everything except a tag in an HTML document, then :not() pseudo class is used. This selector is used to exclude css in a tag.

For example, we want to apply property to those elements which do not have any particular class.

WHAT IS UI ELEMENT PSEUDO CLASS SELECTOR

There are some selectors in pseudo class selectors, which are used for the tags of the form. Which is called ui element pseudo class selector or form element pseudo class selector.

ui element pseudo class selectors list :-

Many types of ui selectors have been defined to select different conditions of form elements and give them style, which you will be able to know from the list below.

1. **:focus**
2. **:checked**
3. **:disabled**
4. **:enabled**
5. **:required**
6. **:optional**
7. **:in-range**
8. **:out-of-range**
9. **:read-only**
10. **:read-write**
11. **:valid**
12. **:invalid**
13. **:default**

:focus :-

When the mouse pointer remains on an input field, that state is called focus state. To apply CSS in the input field on focus state, :focus pseudo class is used.

It does not work on checked boxes, radio buttons, and select boxes. But it works on input type submit.

:checked :-

You can select radio buttons or checkboxes through :checked pseudo class. It works when we select radio buttons and check boxes.

:disabled :-

:disabled pseudo class selectors selects those input fields which are in disabled mode and can give them CSS style.

:enabled :-

:enabled pseudo class selectors select input fields that are enabled.

:required :-

:required pseudo class selectors selects those input fields which are required to be filled i.e. mandatory fields and applies CSS to them.

:optional :-

:optional pseudo class selectors selects those input fields which are optional fields, and applies CSS to them.

:in-range :-

If you want to apply CSS on the basis of some condition in input fields containing numbers, then :in-range pseudo class is used.

:out-of-range :-

If some condition is going wrong in the number input fields, or the user's input is not matching with the range specified, then CSS is applied through :out-of-range pseudo class to tell the user.

:read-only :-

There are some input fields in the form whose value can be read, but no value can be written in it. Such input field is called read-only. By selecting those read-only input fields from the :read-only pseudo class, CSS is applied to them.

:read-write :-

:valid :-

:invalid :-