Lesson:

Understanding the layout and spacing

PW SKILLS

Topics to be covered

1. Width & height
2. Padding and Margins
3. Display Modes

Width & Height

To apply width and height using tailwind classes, we can use w for width and h for height

Syntax:

```
// for width
.w-{size}

// for height
.h-{size}
```

Sizes	
0,1,2,3,4,5,6,	+1
8,10,12,	+2
16,20,24,	+4
32,40,48,56,64	+8
½... 1/{3,4,5,6,12}	Every digit in fraction
screen, full	-

The above table represent the 3 different categories of Sizes,
The 1st one is explicit sizes. 0, 1, 2 , 3, 4 etc... where 4 is 1 rem.
We can also have percentages represented as fractions like ½, ⅓ , ¼ , ⅙ etc where ½ is 50%
We also have screen and full

Let's look into all those categories individually

Let's try to do the one which we did using vanilla css

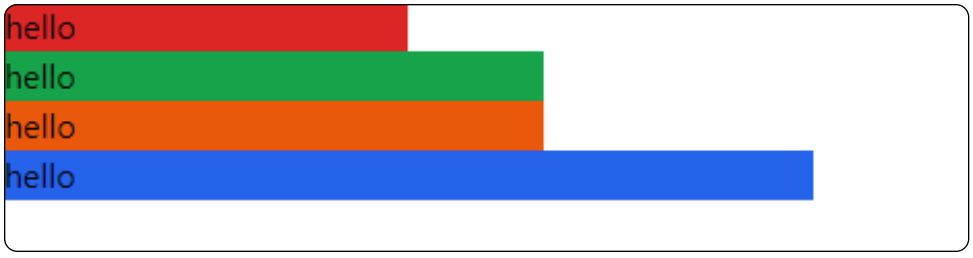
```
<!DOCTYPE html>
<html>
<head>
 <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div class="bg-red-600 w-80 h-80"></div>
</body>
</html>
```

Result:

We have achieved the same using tailwind css. As you can see, it's very easy to use and remember the tailwind classes.
We can directly apply styles in html just by adding the appropriate classes without creating any other css file or codes

Now let's try the percentage width

```
<!DOCTYPE html>
<html>
<head>
 <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div class="bg-red-600 w-3/12">hello</div>
<div class="bg-green-600 w-4/12"> hello</div>
<div class="bg-orange-600 w-4/12">hello</div>
<div class="bg-blue-600 w-6/12">hello</div>
</body>
</html>
```



As you can see in the result, first div is taking 3/12 width, second and third div is taking 4/12 width and the last div is taking 6/12 width which is 50%

w-full v/s w-screen
w-full is nothing but width:100%; in vanilla css while w-screen means width:100vw; in vanilla css

Padding and Margins

To apply padding or margin using tailwind classes, it's very easy

Syntax:

```
// for padding
.p-{size}
// for margin
.m-{size}
```

Sizes	
0,1,2,3,4,5,6,	+1
8,10,12,	+2
16,20,24,	+4
32,40,48,56,64	+8

Example:

```
<!DOCTYPE html>
<html>
<head>
 <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>

  <div class="bg-orange-600 w-32 h-32 m-4">I have got
some margin</div>
<div class="bg-green-600 w-32 h-32 p-4"> i have got some padding</div>

</body>
</html>
```

Result:

In the above example, for the first div, we have applied width and height of 8 rem(128 px), an orange background color and margin of 1rem (1 rem = 4 in tailwind) and for the second div, width and height of 8 rem(128 px), a green background color and padding of 1rem (1 rem = 4 in tailwind)

If we just want to give padding or margin on any one side or either on x or on y axis

Syntax:

```
//padding top
.pt-{size}

//padding bottom
.pb-{size}

//padding right
.pr-{size}

//padding left
.pl-{size}

//padding on x axis
.px-{size}

// padding on y axis
.py-{size}

//margin top
.mt-{size}
```

```
//margin bottom
.mb-{size}

//margin right
.mr-{size}

//margin left
.ml-{size}

//margin on x axis
.mx-{size}

// margin on y axis
.my-{size}
```

Display Modes

Lets now dive into the display modes in tailwind

Syntax:

```
.{display}
```

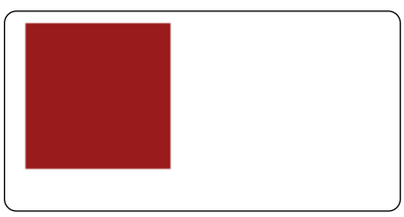
Display Modes	
Block	No widths
inline	
inline-block	
flex	
inline-flex	
table	
table-row	
table-cell	
hidden	

the bold text display modes in the table indicates the commonly and most used display modes

Example:

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
        <div class='m-4 w-20 h-20 bg-red-800'></div>
        <div class='m-4 w-20 h-20 bg-red-800 hidden'></div>
</body>
</html>
```

Result:



In the above example, there are two divs, but in the result you can see only one.Thats Because we have added hidden class in the second div

Flexbox

Flex is one of the important topics in Tailwind CSS, flexbox is tailwind is really easy to use and work with. said in the display modes topic, to apply make a div flex, we just need to add class name flex.
Now let's go a little deeper

Syntax:

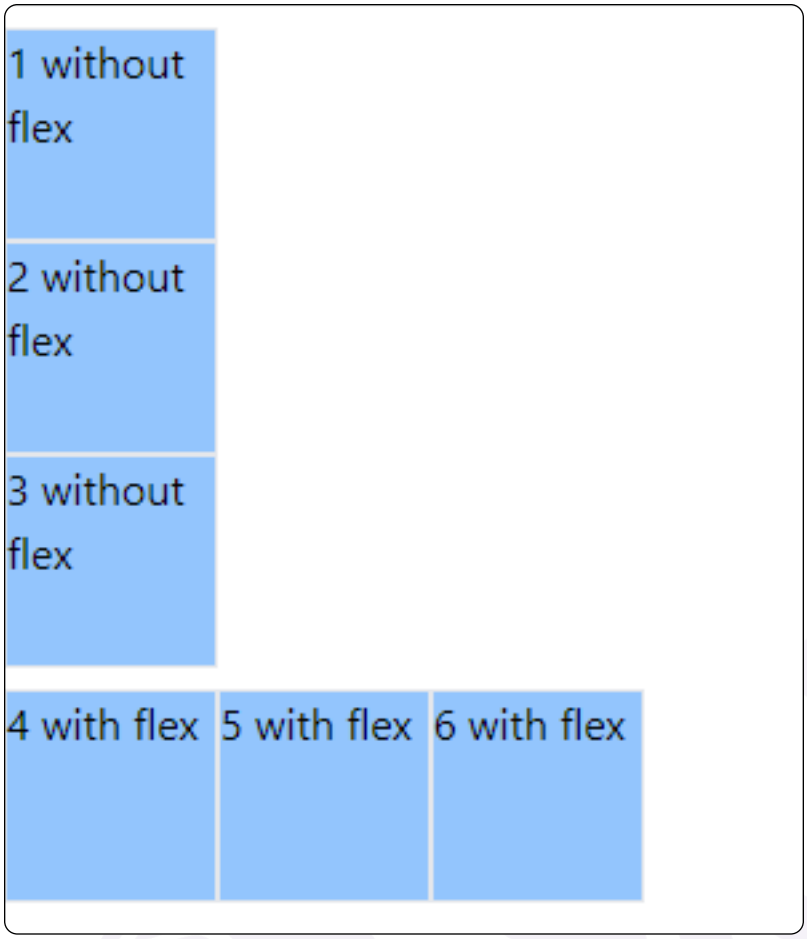
```
.flex
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div class='m-2'>
         <div class='w-20 h-20 bg-blue-300 border'>1 without flex</div>
         <div class='w-20 h-20 bg-blue-300 border'>2 without flex</div>
         <div class='w-20 h-20 bg-blue-300 border'>3 without flex</div>
        </div>

  <div class='m-2 flex'>
         <div class='w-20 h-20 bg-blue-300 border'>4 with flex</div>
         <div class='w-20 h-20 bg-blue-300 border'>5 with flex</div>
         <div class='w-20 h-20 bg-blue-300 border'>6 with flex</div>
        </div>
</body>
</html>
```

Result:



As you can see in the above example,
The first 3 divs are children of a div which was no added flex class
But for the parent of 4, 5, and 6 divs, class flex is added

Flexbox - Justify

Now let's look how to add justify-content property of a flexbox using tailwind css

Syntax:

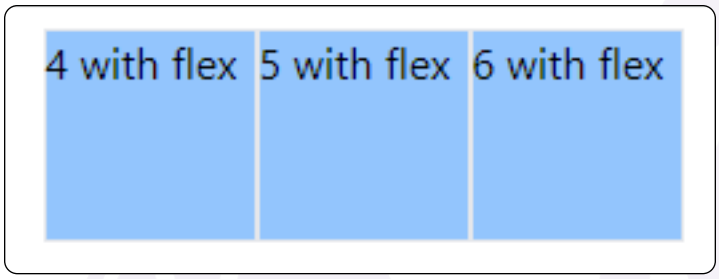
```
.justify-{alignment}
```

Alignments	
start	Default
center	
end	
between	
around	

Example:

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div class='m-2 flex justify-center'>
         <div class='w-20 h-20 bg-blue-300 border'>4 with flex</div>
         <div class='w-20 h-20 bg-blue-300 border'>5 with flex</div>
         <div class='w-20 h-20 bg-blue-300 border'>6 with flex</div>
        </div>
</body>
</html>
```

Result:



As you can see in the above example,
We have added justify-center class for the parent of 4,5,and 6 divs

Flexbox - Align-items

Lets now see how align-items property is added to the flex box using tailwind css

Syntax:

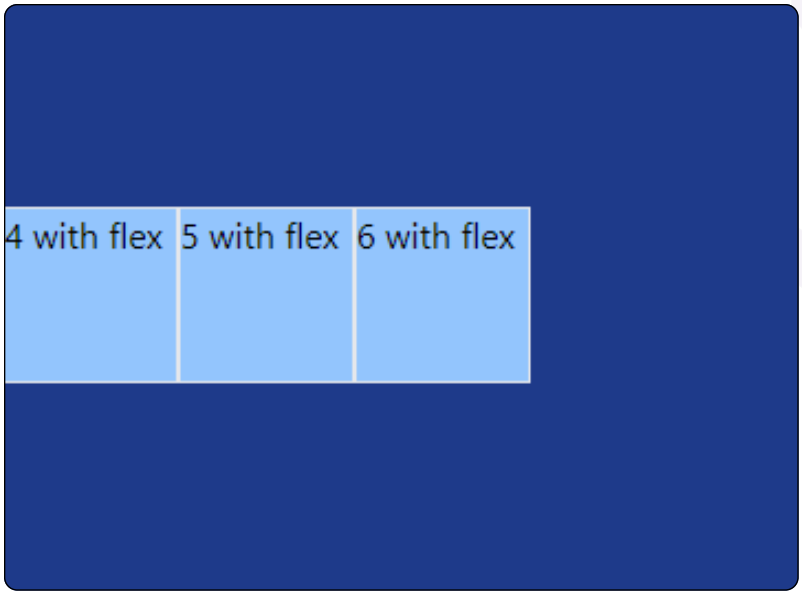
```
.items-{alignment}
```

Alignments	
stretch	Default - fill container
start	
center	
end	
baseline	

Example:

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div class='m-2 flex items-center h-96 bg-blue-900'>
         <div class='w-20 h-20 bg-blue-300 border'>4 with flex</div>
         <div class='w-20 h-20 bg-blue-300 border'>5 with flex</div>
         <div class='w-20 h-20 bg-blue-300 border'>6 with flex</div>
        </div>
</body>
</html>
```

Result:



We have applied a dark blue background color and a height for the parent div and added items-center class to it and the result is as you can see above. The small divs are vertically centred
Similarly you can add other class names in the table as well

Flexbox Direction

There are predefined class names for the flexbox direction property as well

Syntax:

```
.flex-{direction}
```

Directions	
row	Default
row-reverse	
col	
col-reverse	

Example:

```
<div class="border m-2">
      <div>flex-row</div>
      <div class="m-2 flex flex-row">
        <div class="w-20 h-20 bg-blue-300 border">1 with flex</div>
        <div class="w-20 h-20 bg-blue-300 border">2 with flex</div>
        <div class="w-20 h-20 bg-blue-300 border">3 with flex</div>
      </div>
    </div>
    <div class="border m-2">
      <div>flex-col</div>
      <div class="m-2 flex flex-col">
        <div class="w-20 h-20 bg-blue-300 border">1 with flex</div>
        <div class="w-20 h-20 bg-blue-300 border">2 with flex</div>
        <div class="w-20 h-20 bg-blue-300 border">3 with flex</div>
      </div>
    </div>
    <div class="border m-2">
      <div>flex-row-reverse</div>
      <div class="m-2 flex flex-row-reverse">
        <div class="w-20 h-20 bg-blue-300 border">1 with flex</div>
        <div class="w-20 h-20 bg-blue-300 border">2 with flex</div>
        <div class="w-20 h-20 bg-blue-300 border">3 with flex</div>
      </div>
    </div>
    <div class="border m-2">
      <div>flex-col-reverse</div>
      <div class="m-2 flex flex-col-reverse">
        <div class="w-20 h-20 bg-blue-300 border">1 with flex</div>
        <div class="w-20 h-20 bg-blue-300 border">2 with flex</div>
        <div class="w-20 h-20 bg-blue-300 border">3 with flex</div>
      </div>
    </div>
```

Result:

Flexbox Wrap

We can also use the flex wrap property by just adding this tailwind class

Syntax:

```
.flex-{wrap}
```

Wrap modes	
no-wrap	Default
wrap	
wrap-reverse	

Example:

```
<div class="border">
      <div>without flex-wrap</div>
      <div class="m-2 flex">
        <div class="w-24 h-24 bg-blue-300 border">1</div>
        <div class="w-24 h-24 bg-blue-300 border">2</div>
        <div class="w-24 h-24 bg-blue-300 border">3</div>
        <div class="w-24 h-24 bg-blue-300 border">4</div>
        <div class="w-24 h-24 bg-blue-300 border">5</div>
        <div class="w-24 h-24 bg-blue-300 border">6</div>
        <div class="w-24 h-24 bg-blue-300 border">7</div>
        <div class="w-24 h-24 bg-blue-300 border">8</div>
        <div class="w-24 h-24 bg-blue-300 border">9</div>
        <div class="w-24 h-24 bg-blue-300 border">10</div>
      </div>
    </div>
    <div class="border">
      <div>with flex-wrap</div>
      <div class="m-2 flex flex-wrap">
        <div class="w-24 h-24 bg-blue-300 border">1</div>
        <div class="w-24 h-24 bg-blue-300 border">2</div>
        <div class="w-24 h-24 bg-blue-300 border">3</div>
        <div class="w-24 h-24 bg-blue-300 border">4</div>
        <div class="w-24 h-24 bg-blue-300 border">5</div>
        <div class="w-24 h-24 bg-blue-300 border">6</div>
        <div class="w-24 h-24 bg-blue-300 border">7</div>
        <div class="w-24 h-24 bg-blue-300 border">8</div>
        <div class="w-24 h-24 bg-blue-300 border">9</div>
        <div class="w-24 h-24 bg-blue-300 border">10</div>
      </div>
    </div>
```

Result: