UI BASICS

CSS - POSITION

CONTENT

- Introduction
- Position – static
- Position – relative
- Position – fixed
- Position – absolute
- Position – sticky
- CSS z-index

POSITION

- The **position** property specifies the type of positioning method used for an element.
- It can help you to manipulate the location of an element.

There are five different position values:

- ***static*** - by default
- ***relative***
- ***fixed***
- ***absolute***
- ***sticky***

- Elements are then positioned using the **top, bottom, left and right** properties.
- However, these properties will not work unless the position property is set first.
- They also work differently depending on the position value.

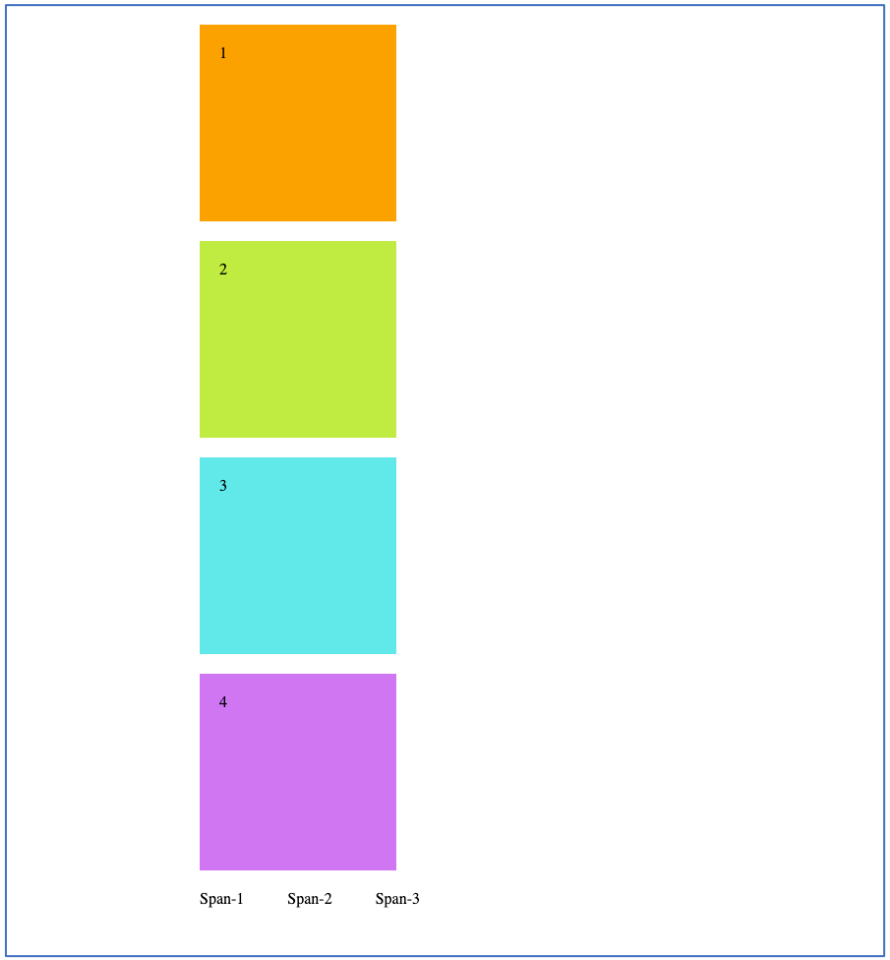
POSITION

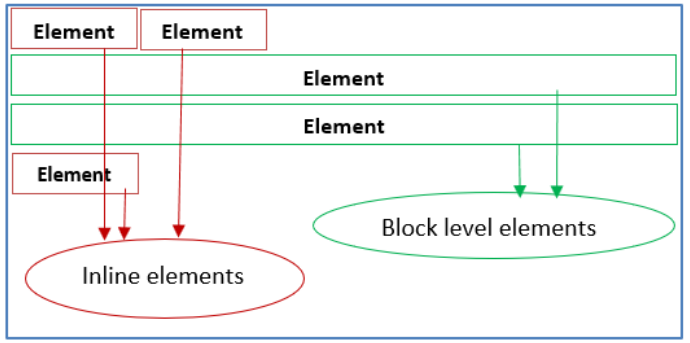
position : static ;

static ------ > every element has a **static** position by default, so the element will stick to the normal page flow.

```css
div {
    width: 200px;
    height: 200px;
    margin: 20px 200px;
    padding: 20px;
    position: static;
}
.one {
    background: orange;
}
.two {
    background: #c3ee40;
}
.three {
    background: #64eaed;
}
.four {
    background: #d079f5;
}
span {
    padding: 20px;
}
span:nth-of-type(1) {
    padding-left: 200px;
}
```

```html
<div class="one">1</div>
<div class="two">2</div>
<div class="three">3</div>
<div class="four">4</div>
<span>Span-1</span>
<span>Span-2</span>
<span>Span-3</span>
```





NOTE : if there is a left/right/top/bottom set then there will be **no effect on that element**.

POSITION

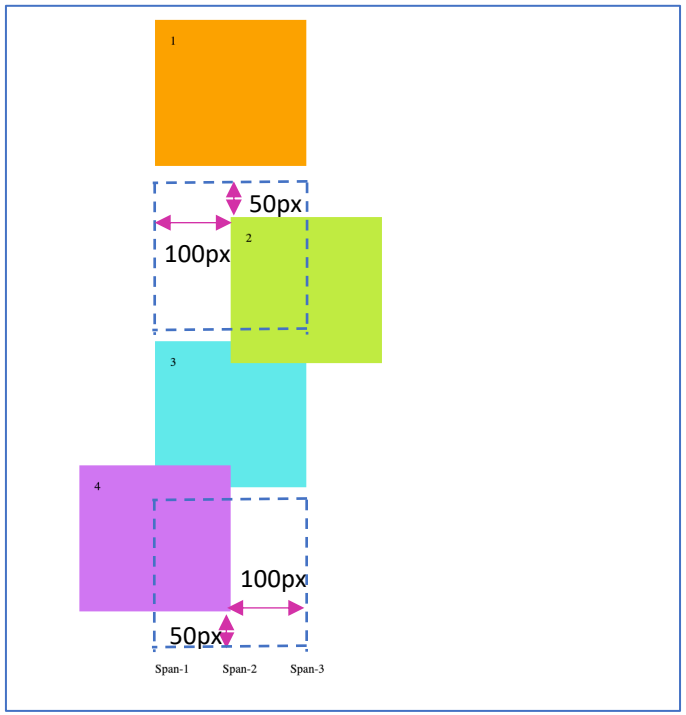
position : relative ; - (*relative to itself*)

relative - an element's **original position remains** in the flow of the document.

- The positional properties "nudge" the element from the original position in that direction.
- It allows element to be offset *relative to* itself.
- The space it takes up doesn't move, so it won't affect anything around it.

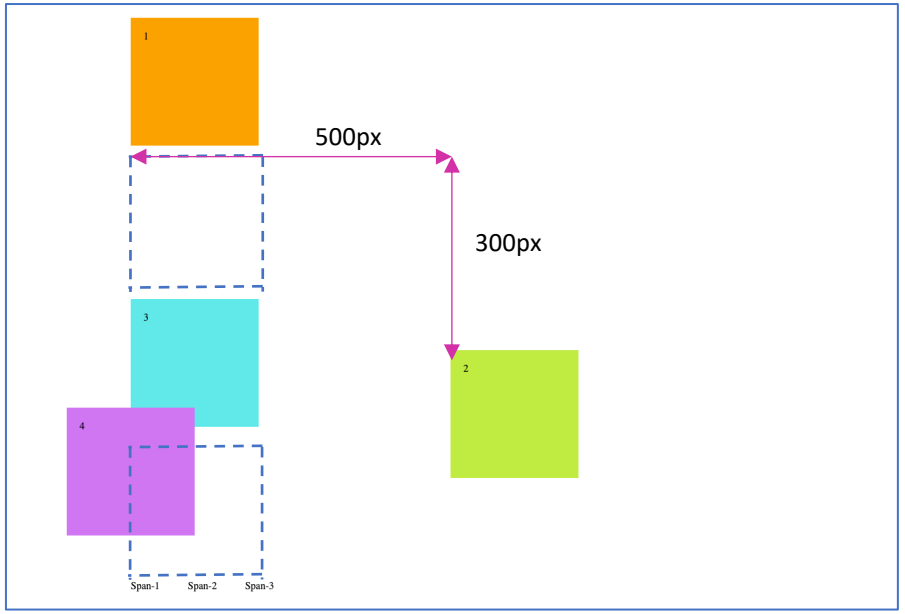
```css
.two {
    background: #c3ee40;
    position: relative;
    top: 50px;
    left: 100px;
}
```

```css
.four {
    background: #d079f5;
    position: relative;
    bottom: 50px;
    right: 100px;
}
```




```css
.two {
    background: #c3ee40;
    position: relative;
    top: 300px;
    left: 500px;
}
```

```css
.four {
    background: #d079f5;
    position: relative;
    bottom: 50px;
    right: 100px;
}
```



NOTE : if there is a left/right/top/bottom set then they **will work.**

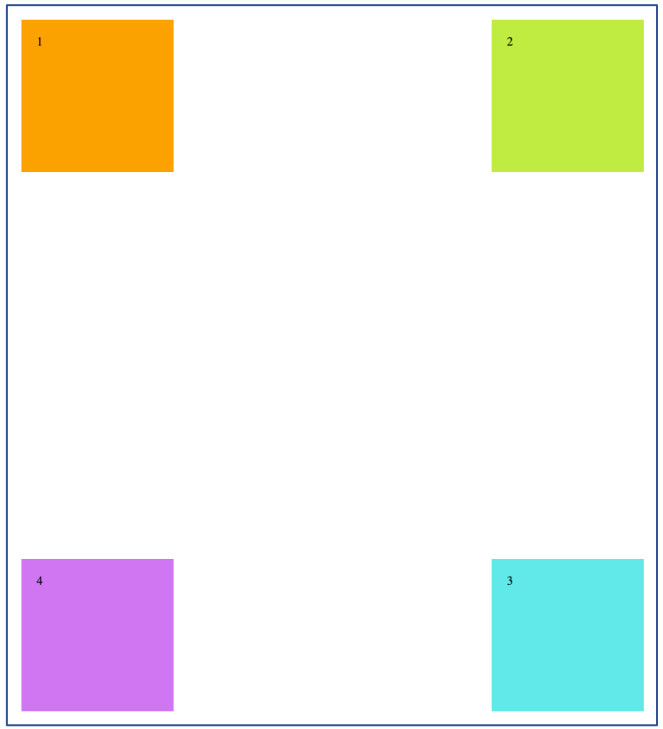
POSITION

position : fixed ; - (*relative to viewport*)

fixed - the element is removed from the flow of the document.

- Fixed positioned elements are always relative to the document/body, **not any particular parent**
- they are unaffected by scrolling. it continues to stick to the position **relative to the document.**

```css
.one {
    background: orange;
    position: fixed;
    top: 0;
    left: 0;
}
.two {
    background: #c3ee40;
    position: fixed;
    top: 0;
    right: 0;
}
.three {
    background: #64eaec;
    position: fixed;
    bottom: 0;
    right: 0;
}
.four {
    background: #d079f5;
    position: fixed;
    bottom: 0;
    left: 0;
}
```



NOTE : if there is a left/right/top/bottom set then they **will work.**

POSITION

Parent : relative;
Child : absolute;

position : absolute ; - (***relative to its nearest parent***)

absolute - The element is taken out of normal flow and leaves ***no space behind***.

- The element is positioned ***relative to its nearest parent*** with a relative-type positioning.

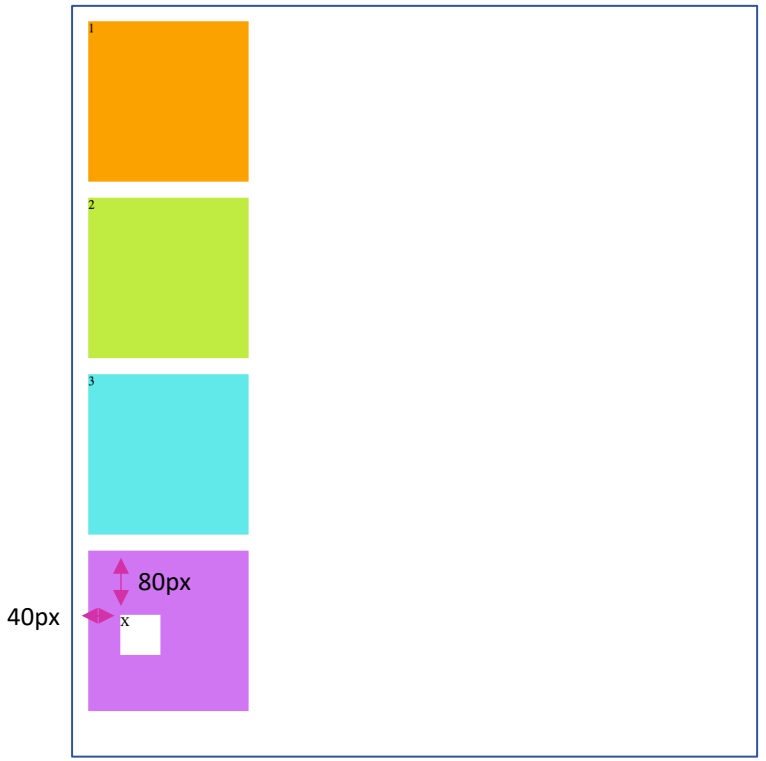
```css
.four {
    background: #d079f5;
    position: relative;
}
.box {
    width: 50px;
    height: 50px;
    background: white;

    position: absolute;
    top: 80px;
    left: 40px;
}
```

Parent {
display: relative;
}

Child {
display: absolute;
}

```html
<div class="one">1</div>
<div class="two">2</div>
<div class="three">3</div>
<div class="four">
    <span class="box">X</span>
</div>
```



NOTE : if there is a left/right/top/bottom set then they **will work.**

POSITION

Parent : relative;
Child : absolute;

position : absolute ; ***- (relative to its nearest parent)***

absolute - The element is taken out of normal flow and leaves ***no space behind***.

- The element is positioned ***relative to its nearest parent*** with a relative-type positioning.

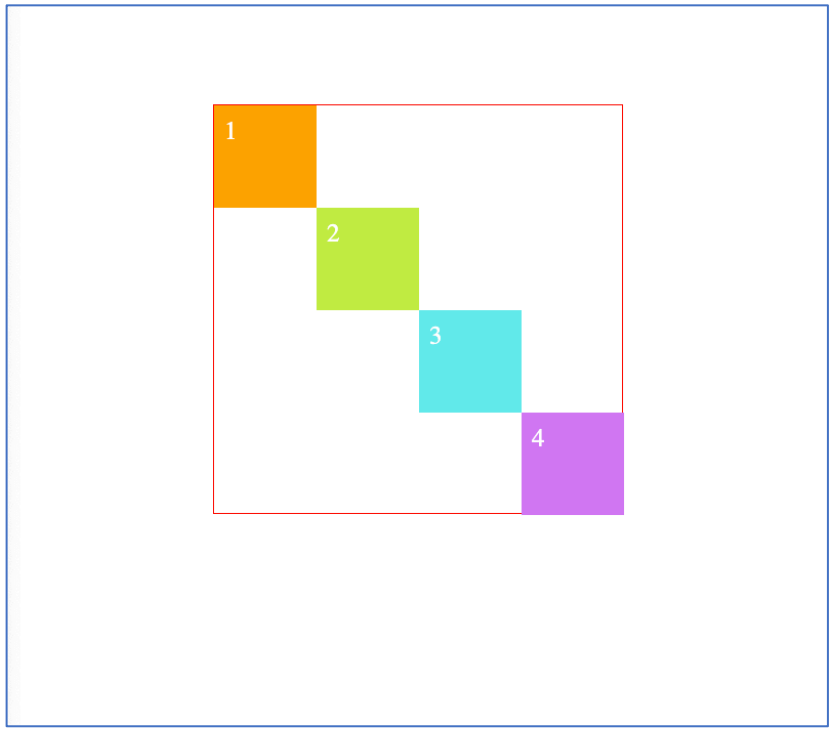
```css
div.container {
    width: 400px;
    height: 400px;
    margin: 100px auto;
    position: relative;
    border: 1px solid red;
}

div {
    width: 100px;
    height: 100px;
    font-size: 25px;
    color: white;
    padding: 10px;
}
.one {
    background: orange;
    position: absolute;
    top:0;
    left: 0;
}
.two {
    background: #c3ee40;
    position: absolute;
    top: 100px;
    left: 100px;
}
.three {
    background: #64eaec;
    position: absolute;
    top: 200px;
    left: 200px;
}
.four {
    background: #d079f5;
    position: absolute;
    top: 300px;
    left: 300px;
```

Parent {
display: relative;
}

Child {
display: absolute;
}

```html
<div class="container">
    <div class="one">1</div>
    <div class="two">2</div>
    <div class="three">3</div>
    <div class="four">4</div>
</div>
```



NOTE : if there is a left/right/top/bottom set then they **will work.**

POSITION

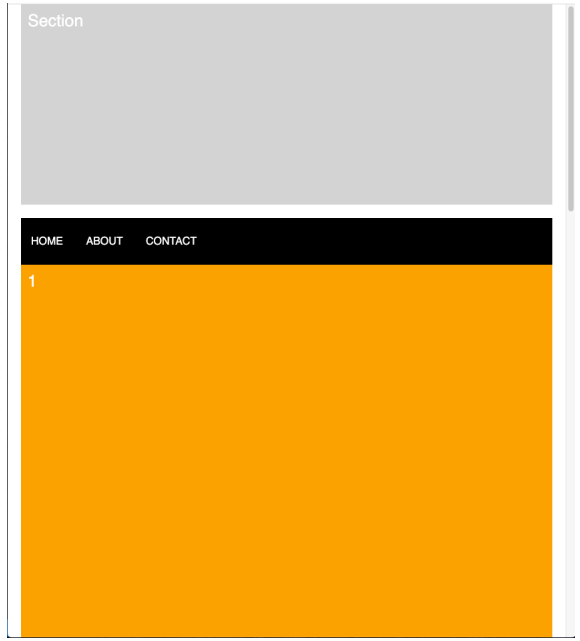
position : sticky ;

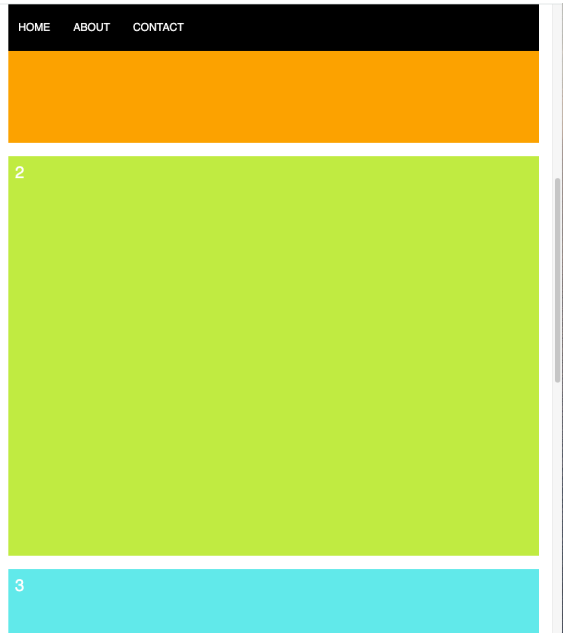
sticky - It allows you to position an element relative to anything on the document and then, once a user has scrolled past a certain point in the viewport, **fix the position of the element to that location so it remains persistently displayed like an element with a fixed value.**

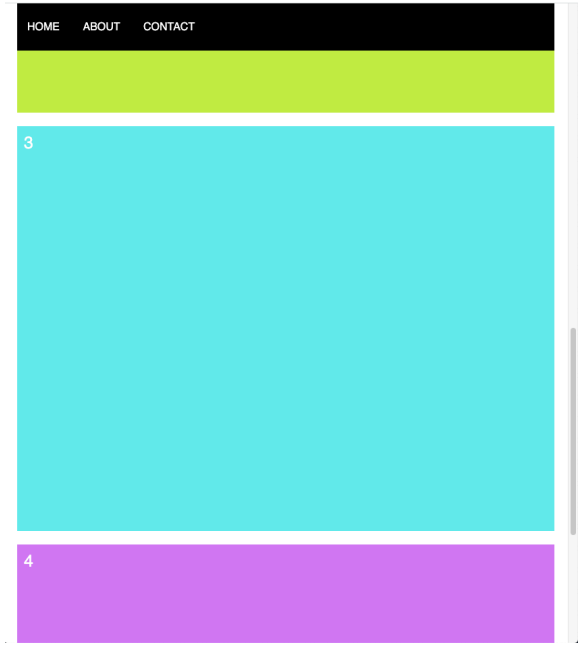
```css
nav {
    width: 100%;
    height: 70px;
    background: black;
    position: sticky;
    top: 0;
}
```

```html
<section>Section</section>
<nav>
    <a href="#">Home</a>
    <a href="#">About</a>
    <a href="#">Contact</a>
</nav>

<div class="one">1</div>
<div class="two">2</div>
<div class="three">3</div>
<div class="four">4</div>
```







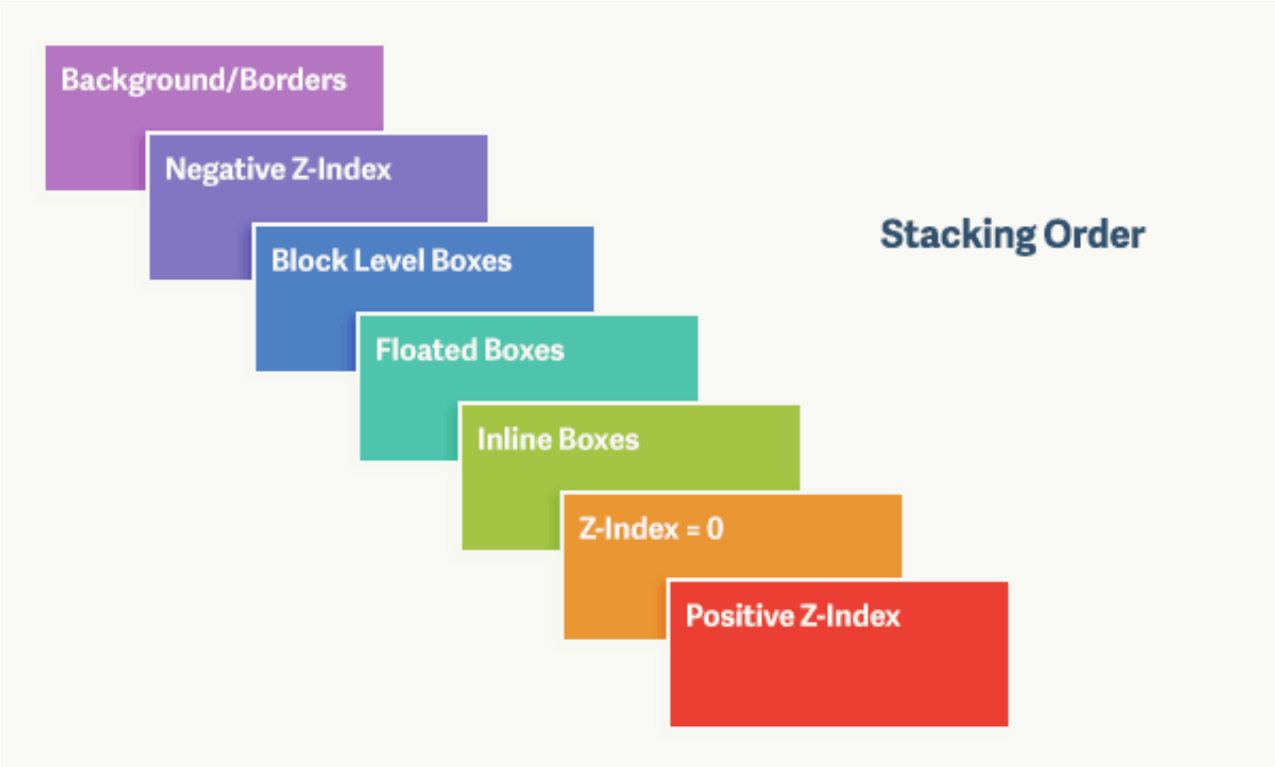
NOTE : if there is a left/right/top/bottom set then they **will work.**

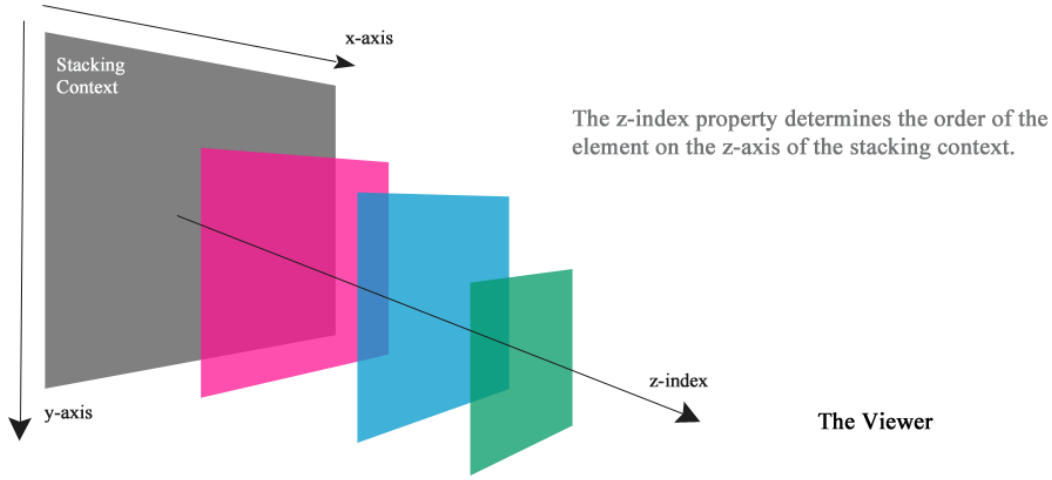
POSITION

CSS z-index

z-index - *It* specifies the stack order of an element.

- An element with greater stack order is always in front of an element with a lower stack order.





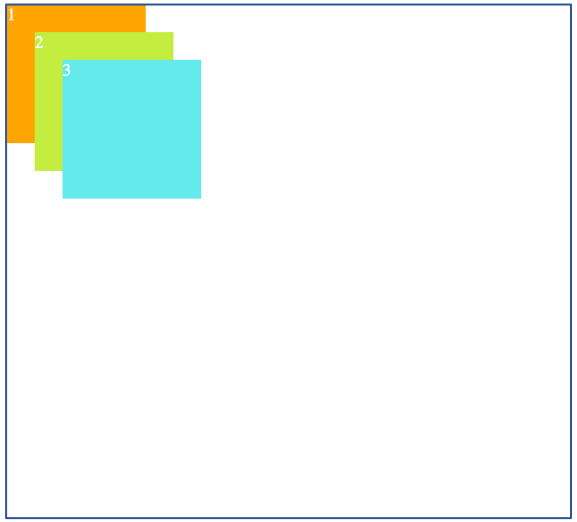
NOTE : z-index only works on positioned elements. (position: absolute, position: relative, position: fixed, or position: sticky)

CSS z-index

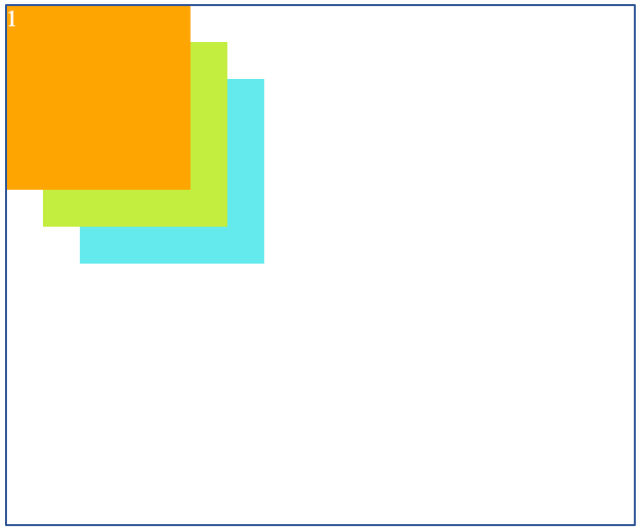
z-index - *It* specifies the stack order of an element.

- An element with greater stack order is always in front of an element with a lower stack order.

```css
div {
    width: 200px;
    height: 200px;
    font-size: 25px;
    color: white;
    position: absolute;
}
.z1 {
    background: orange;
}
.z2 {
    background: #c3ee40;
    top: 40px;
    left: 40px;
}
.z3 {
    background: #64eaec;
    top: 80px;
    left: 80px;
}
```




```css
.z1 {
    background: orange;
    z-index: 2;
}
.z2 {
    background: #c3ee40;
    top: 40px;
    left: 40px;
    z-index: 1;
}
.z3 {
    background: #64eaec;
    top: 80px;
    left: 80px;
}
```




```html
<div class="z1">1</div>
<div class="z2">2</div>
<div class="z3">3</div>
```

NOTE : z-index only works on positioned elements. (*position: absolute, position: relative, position: fixed, or position: sticky*)

POSITION

position: static

- Default positioning for all elements.
- Puts element in normal flow.

position: relative

- Can be offset with **top**, **right**, **bottom** and **left**.
- Offset relative to ***itself***.
- Creates relative-type positioning context for children **(for parents)**.

position: absolute

- Can be offset with **top**, **right**, **bottom** and **left**.
- Offset relative to its ***nearest relative-type positioned parent***.
- Creates relative-type positioning context for children **(for Children).**

position: fixed

- Can be offset with **top**, **right**, **bottom** and **left**.
- Offset relative to ***the viewport***.
- Creates relative-type positioning context for children **(for parents)**.