Positioning in CSS

The CSS properties that allow you to position elements manually in HTML

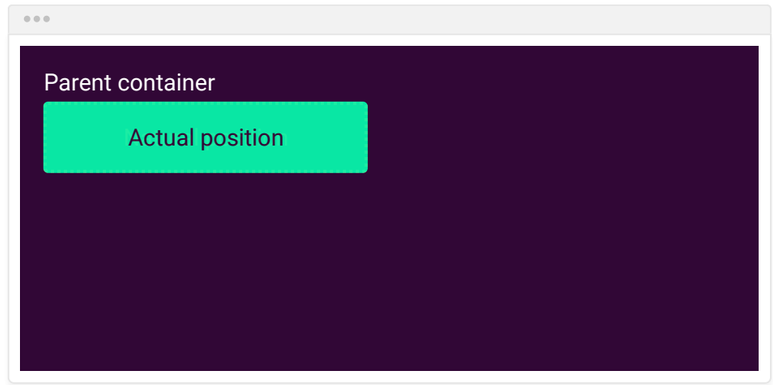
bottom

Defines the position of the element according to its **bottom** edge.

```
bottom: auto;
```

default

The element will remain in its **natural** position.




```
bottom: 20px;
```

If the element is in position **relative**, the element will move *upwards* by the amount defined by the **bottom** value.

```
bottom: 0;
```

If the element is in position **absolute**, the element will position itself from the *bottom* of the first positioned **ancestor**.

left

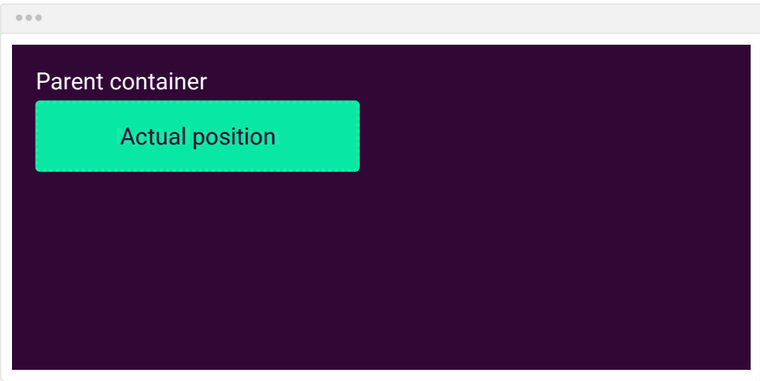
In collection: **Positioning** Permalink Share MDN

Defines the position of the element according to its **left** edge.

```
left: auto;
```

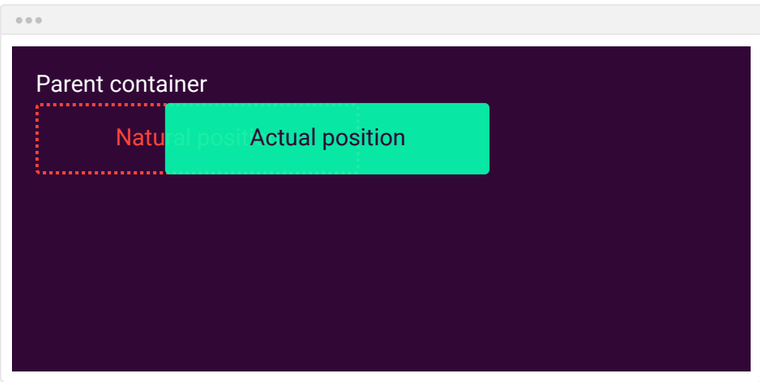
default

The element will remain in its **natural** position.




```
left: 80px;
```

If the element is in position **relative**, the element will move *left* by the amount defined by the **left** value.




```
left: -20px;
```

positioned **ancestor**.

position

In collection: **Positioning** Permalink Share MDN

Defines the position behavior of the element.

`position: static;` default

The element will remain in the **natural flow** of the page.

As a result, it will **not** act as anchor point for the absolutely positioned pink block.

Also, it will **not** react to the following properties:

- `top`
- `bottom`
- `left`
- `right`
- `z-index`

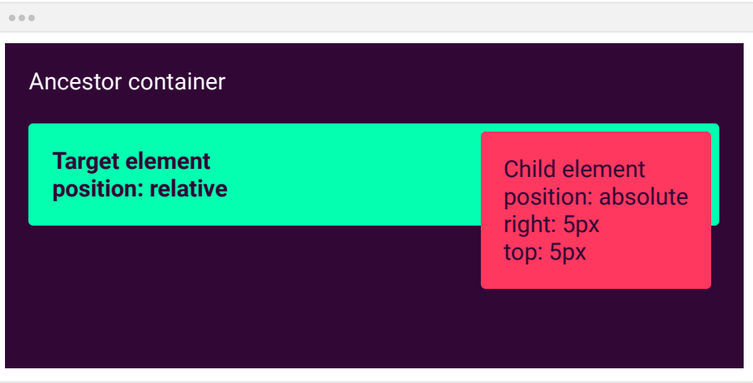
`position: relative;`

The element will remain in the **natural flow** of the page.

It also makes the element **positioned**: it will act as an anchor point for the absolutely positioned pink block.

Also, it will **react** to the following properties:

- `top`
- `bottom`
- `left`
- `right`
- `z-index`

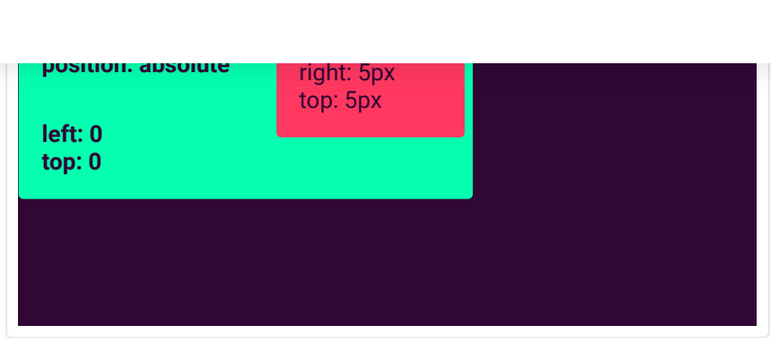


the **closest positioned ancestor**.

Because it's **positioned**, it will act as an anchor point for the absolutely positioned pink block.

Also, it will **react** to the following properties:

- `top`
- `bottom`
- `left`
- `right`
- `z-index`



`position: fixed;`

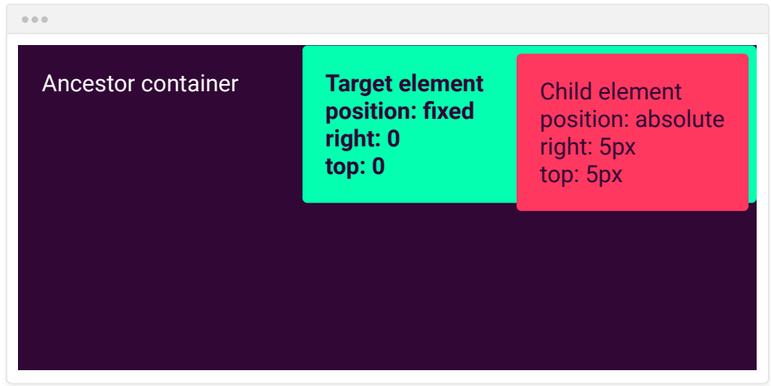
Enable position fixed

The element will **not** remain in the natural flow of the page. It will position itself according to the **viewport**.

Because it's **positioned**, it will act as an anchor point for the absolutely positioned pink block.

Also, it will **react** to the following properties:

- `top`
- `bottom`
- `left`
- `right`
- `z-index`



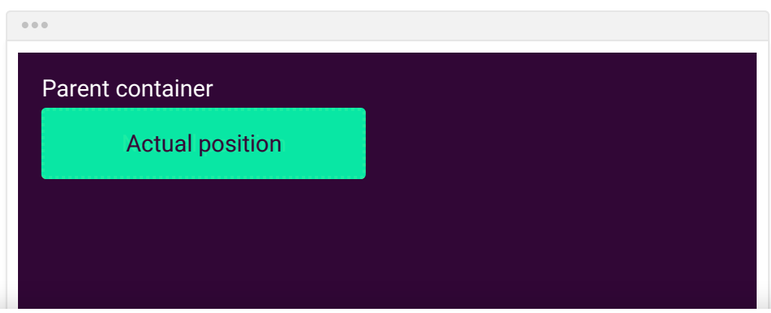
right

In collection: **Positioning** Permalink Share MDN

Defines the position of the element according to its **right** edge.

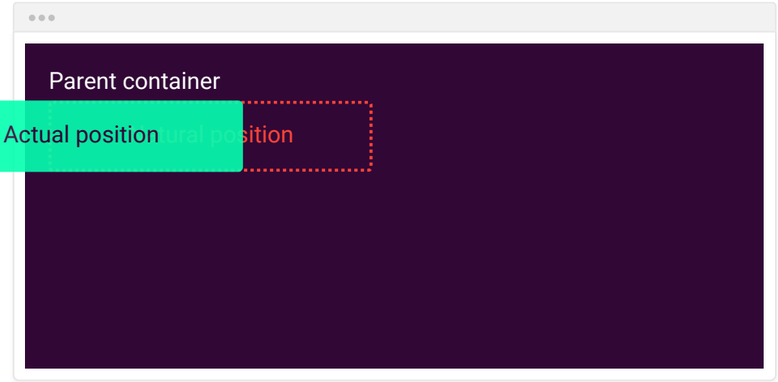
`right: auto;` `default`

The element will remain in its **natural** position.



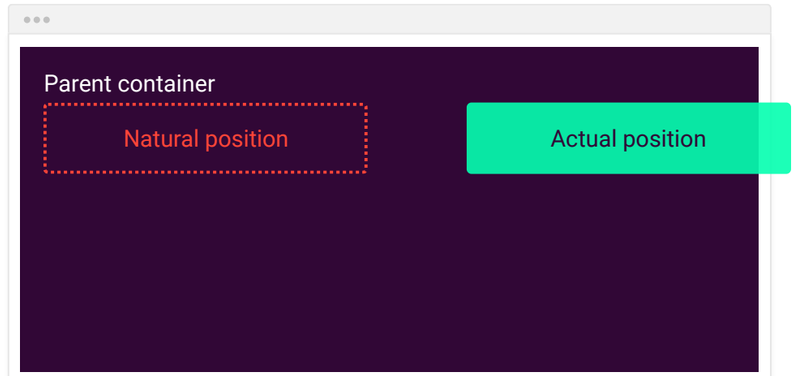
```
right: 80px;
```

If the element is in position **relative**, the element will move *right* by the amount defined by the **right** value.




```
right: -20px;
```

If the element is in position **absolute**, the element will position itself from the *right* of the first positioned **ancestor**.



top

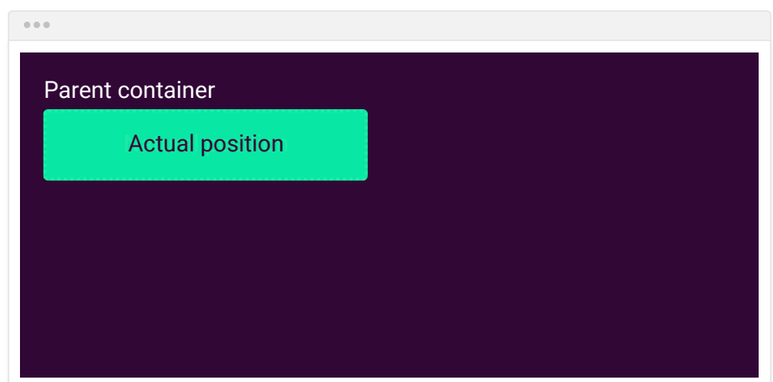
In collection: **Positioning** Permalink Share MDN

Defines the position of the element according to its top edge.

```
top: auto;
```

`default`

The element will remain in its **natural** position.



cssreference.io

All properties Animations
Backgrounds Box model
Flexbox CSS Grid
Positioning Transitions
Typography

Search for a property

bottom
left
position
right
top
z-index

If the element is in position **relative**, the element will move *downwards* by the amount defined by the **top** value.


```
top: 0;
```

If the element is in position **absolute**, the element will position itself from the *top* of the first positioned **ancestor**.

z-index

In collection: **Positioning** Permalink Share MDN

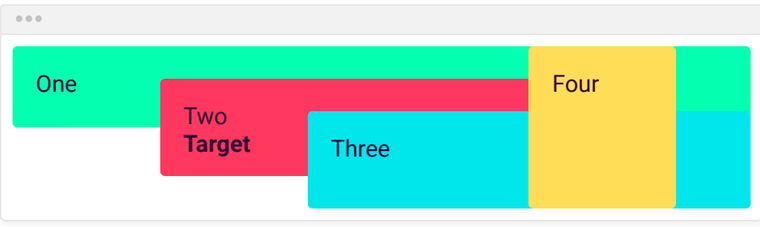
Defines the **order** of the elements on the **z-axis**. It only works on **positpositioned** elements (anything apart from `static`).

```
z-index: auto;
```

default

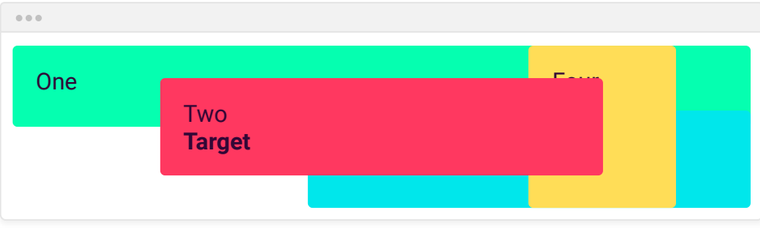
The order is defined by the order in the **HTML code**:

- first in the code = behind
- last in the code = in front




```
z-index: 1;
```

The z-index value is **relative** to the other ones. The target element is move in **front** of its siblings.



You can use **negative values**. The target element is move in **behind** its siblings.

cssreference.io

All properties Animations
Backgrounds Box model
Flexbox CSS Grid
Positioning Transitions
Typography

Search for a property

bottom
left
position
right
top
z-index

codecademy

Display and Positioning

CSS `z-index` property

The CSS `z-index` property specifies how far back or how far forward an element will appear on a web page when it overlaps other elements.

The `z-index` property uses integer values, which can be positive or negative values. The element with the highest `z-index` value will be at the foreground, while the element with the lowest `z-index` value will be at the back.

```
//`element1` will overlap `element2`
.element1 {
  position: absolute;
  z-index: 1;
}

.element2 {
  position: absolute;
  z-index: -1;
}
```

Fixed CSS Positioning

Positioning in CSS provides designers and developers options for positioning HTML elements on a web page.

The CSS `position` can be set to `static`, `relative`, `absolute` or `fixed`. When the CSS position has a value of `fixed`, it is set/pinned to a specific spot on a page. The fixed element stays the same regardless of scrolling. The navigation bar is a great example of an element that is often set to `position:fixed;`, enabling the user to scroll through the web page and still access the navigation bar.

```
navbar {
postion : fixed;
 }
```

CSS `display` property

The CSS `display` property determines the type of render block for an element. The most common values for this property are `block`, `inline`, and `inline-block`.

Block-level elements take up the full width of their container with line breaks before and after, and can have their height and width manually adjusted.

Inline elements take up as little space as possible, flow horizontally, and cannot have their width or height manually adjusted.

Inline-block elements can appear next to each other, and can have their width and height manually adjusted.

```
.container1 {
  display: block;
}

.container2 {
  display: inline;
}

.container3 {
  display: inline-block;
}
```

CSS `position: absolute`

The value `absolute` for the CSS property `position` enables an element to ignore sibling elements and instead be positioned relative to its closest parent element that is positioned with `relative` or `absolute`. The `absolute` value removes an element entirely from the document flow. By using the positioning attributes `top`, `left`, `bottom` and `right`, an element can be positioned anywhere as expected.

```
.element {
  position: absolute;
}
```

CSS `position: relative`

The value `relative` of the CSS `position` property enables an element to be positioned relative to where it would have originally been on a web page. The offset properties can be used to determine the actual position of the element relative to its original position. Without the offset properties, this declaration will have no effect on its positioning, it will act as the default value `static` of the `position` property.

```
.element {
  position: relative;
}
```

CSS `float` property

The CSS `float` property determines how far left or how far right an element should float within its parent element. The value `left` floats an element to the left side of its container and the value `right` floats an element to the right side of its container. For the property `float`, the `width` of the container must be specified or the element will assume the full width of its containing element.

```
/* The content will float to the left
side of the container. */
.left {
  float: left;
}

/* The content will float to the right
side of the container. */
.right {
  float: right;
}
```

The CSS `clear` property

The CSS `clear` property specifies how an element should behave when it bumps into another element within the same containing element.The `clear` is usually used in combination with elements having the CSS `float` property. This determines on which sides floating elements are allowed to float.

```
/*This determines that no other elements within the same containing element are allowed to float on the left side of this element.*/
.element {
  clear: left;
}

/*This determines that no other elements within the same containing element are allowed to float on the right side of this element.*/
.element {
  clear: right;
}

/*This determines that no elements within the same containing element are allowed to float on either side of this element.*/
.element {
  clear: both;
}

/*This determines that other elements within the same containing element are allowed to float on both side of this element.*/
.element {
  clear: none;
}
```