position:  
  fixed  
  static  
  relative  
  absolute
Recall – Alternative Element Positions

Floating elements is an alternative to normal document flow.

Let’s look at positioning schemes

Relative and absolute

Allow designer to specify precise offsets

Move target element to different page origins.
Static vs relative positioning schemes

The `position` property specifies the type of positioning method used for an element (static, relative, fixed or absolute).

The `position` Property

The `position` property specifies the type of positioning method used for an element.

There are four different position values:

- `static`
- `relative`
- `fixed`
- `absolute`

Static positioning is a different way of saying normal document flow

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: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

This `<div>` element has position: static;
• **Static.** This is the default for every single page element. Different elements don't have different default values for positioning, they all start out as static. Static doesn't mean much, it just means that the element will flow into the page as it normally would. The only reason you would ever set an element to position: static is to forcefully-remove some positioning that got applied to an element outside of your control. This is fairly rare, as positioning doesn't cascade.

https://css-tricks.com/absolute-relative-fixed-positioning-how-do-they-differ/
Static positioning

- Normal document flow
- Default for all elements, except `html`
- If you set the position to static:
- Any positioning offsets you use will be ignored
position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

This `<div>` element has position: relative;

When you apply position relative on an element. The element is positioned relative to its position in the normal document flow.
**Relative.** This type of positioning is probably the most confusing and misused. What it really means is "relative to itself". If you set `position: relative;` on an element but no other positioning attributes (top, left, bottom or right), it will no effect on it's positioning at all, it will be exactly as it would be if you left it as position: static; But if you DO give it some other positioning attribute, say, `top: 10px;`, it will shift it's position 10 pixels DOWN from where it would NORMALLY be. I'm sure you can imagine, the ability to shift an element around based on it's regular position is pretty useful. I find myself using this to line up form elements many times that have a tendency to not want to line up how I want them to.

There are two other things that happen when you set position: relative; on an element that you should be aware of. One is that it introduces the ability to use z-index on that element, which doesn't really work with statically positioned elements. Even if you don't set a z-index value, this element will now appear **on top** of any other statically positioned element. You can't fight it by setting a higher z-index value on a statically positioned element. The other thing that happens is it **limits the scope of absolutely positioned child elements.** Any element that is a child of the relatively positioned element can be absolutely positioned within that block. This brings up some powerful opportunities which I talk about here.

Relative positioning

When you apply position relative to an element:

The element is positioned relative to its position in the normal document flow.

If you were to apply offsets of that element, the offsets will be applied relative to the original document flow position.
Flow and anchoring...

When you position an element as “relative” you are creating an anchor for the offset.

The element’s “top” “bottom” “left” and “right” edges become the boundary anchors from which you can offset the element.
Important:

Element is NOT taken out of normal document flow!!!

Even if the element is *moved* (due to the offset), its original spot is preserved.

The element is *visually in a different spot*, but as far as all of the elements in the HTML page, the element is still sitting in its original spot.
Example: relative positioning

This paragraph is represented by this orange box

Another element in the document
Example: relative positioning

As soon as one sets the position to relative, the edges on the paragraph become an anchor for future offsets.
From the top, move the element 50px

top: 50px;

From the left, move the element 50px

left: 50px;

Note: this element’s positioning in the document was unaffected by the offset.

When the offsets are applied, the element moves relative to the normal document flow – that is its original position.

top/left

nothing really changes in the rest of the document

Another element in the document
Another element in the document

bottom/right
Example to show relative positioning:

<table>
<thead>
<tr>
<th>Let's start positioning elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Red Square]</td>
</tr>
</tbody>
</table>
Let's start positioning elements

```
<style>
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

h1 {
  margin-bottom: 15px;
}

div#container {
  background-color: #cccccc;
}

p {
  width: 50px;
  height: 50px;
  border: 1px solid black;
  margin-bottom: 15px;
}

#p1 {
  background-color: #a2a2a2;
}

#p2 {
  background-color: #deb887;
}

#p3 {
  background-color: #5f9ea0;
}

#p4 {
  background-color: #ff7f50;
}
</style>

<body>

<h1>Let's start positioning elements</h1>

<div id="container">
  <p id="p1"></p>
  <p id="p2"></p>
  <p id="p3"></p>
  <p id="p4"></p>
</div>

</body>
```

Here the div container has the familiar light greyish color, and every paragraph tag is 50 pixels by 50 pixels, they have a black border of one pixel, and also has a margin bottom of 15 pixels.
For p1, we will add a `position: relative;` property.

Save.
Reload page.
Did anything change?

YES! something DID change!!
Added the following offsets.

What do you predict will happen when the page is reloaded?

It is set top: 65px, so it's **from** the top, 65 pixels down.
And then **from** the left, 65 pixels from the left.
The other elements have not moved.

Let's start... Elements

The original spot is still vacant.

Element is NOT taken out of the document flow.
Absolute positioning: all offsets, top, bottom, left, right, are all *relative* to the position of the nearest ancestor (e.g. parent or grandparent) which has positioning set on it other than static.

```
position: absolute;
```

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However, if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

**Note:** A "positioned" element is one whose position is anything except `static`.

By default, HTML element is the only element that has non-static positioning set on it.
**Absolute.** This is a very powerful type of positioning that allows you to literally place any page element exactly where you want it. You use the positioning attributes top, left bottom and right to set the location. Remember that these values will be relative to the next parent element with relative (or absolute) positioning. If there is no such parent, it will default all the way back up to the `<html>` element itself meaning it will be placed relatively to the page itself.

The trade-off, and most important thing to remember, about absolute positioning is that these elements are removed from the flow of elements on the page. An element with this type of positioning is not affected by other elements and it doesn't affect other elements. This is a serious thing to consider every time you use absolute positioning. It's overuse or improper use can limit the flexibility of your site.

Now for **absolute** positioning

How?

Want to move p3 to the position of p1.
Set its position to absolute position top: 0 and left: 0
1. The element was taken out of the document flow.
2. The top-left edge of the element is now flush against the edge of the browser.
3. Not where we wanted it.
Recall: absolute positioning needs an absolute or relative parent or ancestor.
Recall: absolute positioning needs an absolute or relative parent or ancestor.

Let's start positioning elements.

Does `<body>`, parent of `<div>` have its position set to either relative or absolute?

**No.**
Does `<html>` have its position set to either "relative" or "absolute" by default?
we just need to anchor the ancestor or the container of our paragraph tag with the position relative. Any element contained within that container element will then be positioned relative to its edges.
move the entire container element down and see if all of the elements within it will move down and stay in the same arrangement

```css
div#container {
  background-color: #cacaca;
  position: relative;
  top: 60px;
}
```
Remember:

- Static positioning is default for all elements, except `html`
- Relative positioning offsets the element relative to its normal document flow position
- Absolute positioning is relative to closest ancestor which has positioning set to non-static value
- Offsetting the relative container element offsets its contents as well