CSS Dropdown Menus

A dropdown menu is a common component in a web page where you want to organize a menu so that the initial visible menu is neater and less cluttered with links. Upon hovering on a menu link with more sub-links, a submenu appears. To do this with CSS only, it is first important to understand the CSS position property which is the key to a CSS dropdown. (Note that you can implement a dropdown to expand upon click, but that is an exercise for Javascript.)

Let's take a closer look at the CSS position property before we look at the code for a dropdown.

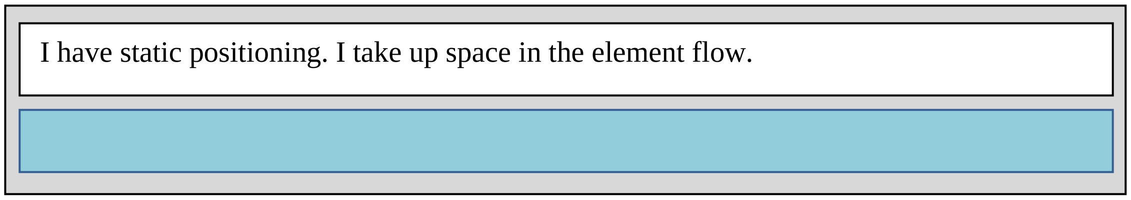
Position property

The position property is an essential property to understand. It is useful to position something relative to other elements or the window. In most cases, using flex or grid or margin/padding, you won't need the position property. Where it really shines is place elements relative to the browser window or when you need small adjustments (e.g. CSS artwork). The values below are commonly used.

`position:` *[static/relative/absolute/fixed]*`;`

static

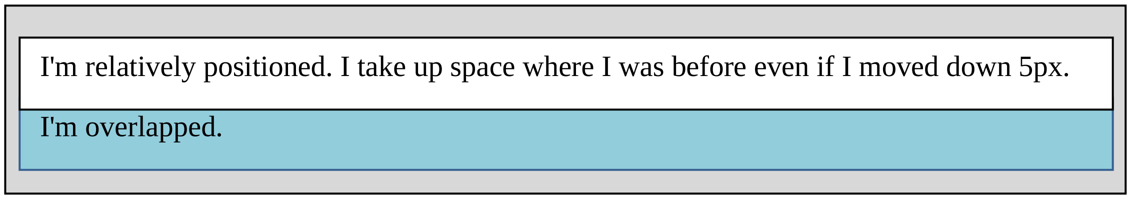
This is the **default positioning for all elements**. Elements with static positioning show up in their natural locations and take up space in the flow of elements. Elements with static positioning **<u>cannot</u>** be moved using the **top**, **left**, **right**, or **bottom** properties.



I have static positioning. I take up space in the element flow.

relative

Elements with relative positioning also show up in their natural locations by default and also take up space in the flow of elements. The difference between static positioning and relative positioning is that elements with relative positioning **<u>can</u>** be moved using the **top**, **left**, **right**, or **bottom** properties. Although relatively positioned elements can be moved, they still take up space where they were originally located. **<u>When you move an element with relative position, the element is moved relative to its default position.</u>**



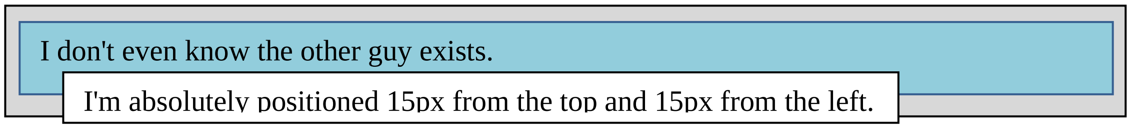
I'm relatively positioned. I take up space where I was before even if I moved down 5px.

I'm overlapped.

absolute

Elements with absolute positioning are taken out of the normal flow of elements. Other elements no longer leave space where the element should have been located. Absolutely positioned elements are positioned **relative to the closest parent with non-static positioning**. If no parent exists with non-static positioning, the element will be positioned relative to the body/document (`<body>` element). Block elements which are absolutely positioned lose their full widths and shrink to their content's widths because they are taken out of the normal flow.

Below, the parent container has relative positioning.


I don't even know the other guy exists.
I'm absolutely positioned 15px from the top and 15px from the left.

fixed

Elements with fixed positioning are also taken out of the normal flow of elements (so other elements do not leave space for them), but they are positioned **relative to the window**. This allows you to permanently affix an element in the window's view. This is how a "sticky" menu bar at the top of the window is achieved.

Changing locations:

You can change where an element is located using the **top**, **left**, **right**, and/or **bottom** properties *as long as it does not have static positioning* (e.g. the position property needs to be set to either relative, absolute, or fixed).

The **top**, **left**, **right**, and **bottom** properties allows you to specify a length or distance from the given edge of an element. For example, specifying

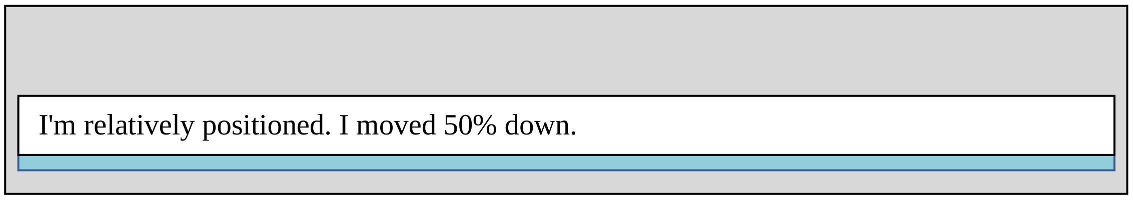
```
top: 10px;
```

means to move 10px down from the top. These distances do not need to be static values (e.g. 10px) but can also be relative (e.g. 10%). Be aware that when using relative values like percentages, the distance is calculated using the parent's height or width.

For example:

```
position: relative;
top: 50%;
```

will move an element 50% down the parent container from the top.



If you have an absolutely positioned element which is positioned 50% from the top and there is *no* ancestor with non-static positioning, the element will be positioned 50% from the top of the window.

sticky

When an element is using "sticky" as the position, it will initially appear in the natural location (according to the semantic code). Upon scrolling past the element, it will sticky itself relative to the nearest ancestor that can scroll. (This is how you can create a sticky menu.)

CSS dropdown menus

1. Structure your dropdown menu as a **nested lists of links**. (The submenus are the nested lists.)

```
...
<nav id="main-menu" aria-label="Main navigation">
  <ul class="inline-menu">
    <li><a href="#">Link one</a>
      <ul>
        <li><a href="#">Sublink 1</a></li>
        <li><a href="#">Sublink 2</a></li>
      </ul>
    </li>
    <li><a href="#">Link two</a></li>
    <li><a href="#">Link three</a>
      <ul>
        <li><a href="#">Sublink 1</a></li>
     </ul>
    </li>
  </ul>
</nav>
...
```

2. You can set the outer menu to be inline by setting the display property to flex (by default it will show in a row. You can add a little space between list items. Also make sure everything is bunched at the top so that you don't see a stretched out height for inline menu items.

```
.inline-menu {
  display:flex;
  gap:3em;
  align-items:flex-start;
}
```

Display property

The display property can be set to inline, block, inline-block, flex, and grid. Values of "flex" and "grid" set the container to use flexbox and grid, respectively. The other three values are a little less straight-forward.

Inline
All inline elements will display side by side. A display value of "inline" displays like other flow

elements (e.g. span, a). **Elements using an "inline" display value cannot have their widths and/or heights set.**

Block
Display "block" makes an element display like any other block element (default 100% width and on its own line). Blocks can have their widths and heights changed.

Inline-block
An inline-block sets an element to display side-by-side with other inline elements but it also **allows you to change the width and height** like a block. Any space in the HTML (including a new line) will add a space between inline-blocks that cannot be removed by setting margin to 0. The best fix for this is to comment out the extra space in the HTML. Another option is a negative margin but this is less reliable.

3. Make sure the bullet points are gone remove default margin and padding. Since <nav> is used to wrap around major menus, we can just remove bullet points and default margin/padding for any <ul> within a <nav>.

```
nav ul {
  list-style:none;
  margin:0;
  padding:0;
}
```

4. In order to be able to hide the submenu links from view, we need to move them off screen. For better screen reader accessibility, you can keep the submenu links readable by screen readers by not removing them complete. Use absolute positioning on the inner list to move the submenu completely out of view.

```
.inline-menu ul {
  position:absolute;
  left:-10000px;
}
```

   Remember that absolute positioning takes an element out of the normal flow of the page so other elements act as if the absolutely positioned element is not even there. This is perfect for a dropdown because it overlap. (If, for some reason, an absolutely-positioned element gets covered by another element, you can use the **z-index** property to bring the element forward.)
5. Now you want the submenu to show up when you hover over a link in the outer list.

```
#main-menu li:hover > ul {
  left:0;
}
```

6. Check it in the browser. The submenu will show up in a funny spot. This is because the positioning is relative to the window since we didn't set the position property on any ancestors of the inner <ul>. Since we want the submenu position to be appropriately under the link we hover over, set the position for the parent <li>.

```
.inline-menu li {
  position:relative;
}
```

7. The menus should show up properly now. Now complete the styling of your links as you wish.

Consider: How would you deal with a sub-submenu?