## **Positioning & Alignment**

In the last subsection, you learned some very important properties to manipulate HTML elements. Using height, width, margin, border, and padding you’re now able to change the size of elements, adjust the space between the content of the element and the border, and much more. However, one essential thing is still missing – **positioning** of HTML elements.

Two aspects have to be considered when **positioning elements** with CSS:

* The **value** of the position property assigned to the HTML element
* The exact location of the HTML element, adjustable with the CSS properties left, top, right, and bottom

The **location** properties (left, top, right, and bottom) are **dependent** on the position property. In fact, **without** a **position** property, the **location** properties **won’t** **work**. Therefore, if tweaking the location parameters doesn’t yield the desired result, check the position property first.

**Different possibilities for the position value:**

The property **position** can take the values **static** (the default value), **absolute**, **relative**, and **fixed**.

* **Fixed** - stick to the same position and stays visible even if you scroll.
* **Relative** - is located relative to its original position.
* **Absolute** - positioned relative to their nearest ancestor with an explicit position.
* **Static (default)** - revokes the effect of **top**, **right**, **bottom**, and **left** as it is the default value of **position**.
* **Inherit** - like most properties, it can also be told to explicitly **inherit** a value, but it doesn't do so by default.

#### Position: fixed

If you assign the value fixed to the position property of an HTML element, the **location properties** will work relative to the viewport. This means that a fixed element stays at the same position on the screen – even if you scroll. So, it’s your choice for elements that need to be **visible all the time**, for instance a navigation bar in a webshop.

#### Position: relative

If you assign the CSS property position: relative, the element will (still) sit at its original position. However, using the **four location properties** you can now shift the HTML element **relative** to its normal position.

Take a look at the next example where the sky and space switched their original order and where the landscape even escaped the universe…

#### Position: absolute

When assigning the CSS property position: absolute to an element, the parent element and other ancestors become relevant, since the HTML element will be positioned **relative to its nearest ancestor** with an **explicit position**. With **explicitly positioned ancestors** we mean all HTML elements which contain the styled element as a child in any level and have either a fixed , relative or absolute position.

#### Position: static

The **last value** that can be assigned to position is static. However, this is the **default value** of the HTML element, even without explicitly assigning it. The **four location properties** won’t have any effect, when the position is set to static.

If this explanation of the topic left you flummoxed, don’t hesitate to browse through other explanations:

* [W3schools](http://www.w3schools.com/css/css_positioning.asp)
* [BarelyFitz](http://www.barelyfitz.com/screencast/html-training/css/positioning/)
* [aListApart](http://alistapart.com/article/css-positioning-101)

#### Position: inherit

/\* Let the <section>, the <footer>, and the app-bar inherit the body's position with only one rule. \*/

section, footer, #app-bar {

position: inherit;

        }

<!DOCTYPE html>

#### Position: fixed

If you assign the value fixed to the position property of an HTML element, the **location properties** will work relative to the viewport. This means that a fixed element stays at the same position on the screen – even if you scroll.

So it’s your choice for elements that need to be **visible all the time**, for instance a navigation bar in a webshop.

<html>

<head>

  <title>**Position Fixed**</title>

  <style>

      body {

        margin: 0;

        padding: 0;

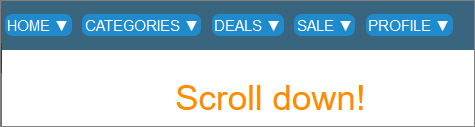
        font-family: sans-serif;

      }

      #task {

        text-align: center;

        font-size: 28px;

        color: darkorange;

        line-height: 100px;

      }

      #nav-bar {

        z-index: 1;

        width: 100%;

        position: fixed;

        top: 0; /\* visible all the time, even if you scroll \*/

        color: white; /\* button text \*/

        text-transform: uppercase;

        background-color: #3A657F; /\* menu strip color \*/

      }

      /\* styling buttons\*/

      div span {

        font-size: 12px;

        line-height: 40px;

        background-color: #208BCC; /\* button background color \*/

        border-radius: 6px; /\* button rounded corners \*/

        padding: 2px; /\* button size \*/

        margin: 2px; /\* space between buttons (inline as it's for <span>) \*/

      }

      #placeholder {

        height: 800px;

        background-color: beige;

      }

</style>

</head>

<body>

      <!--Steady navigation bar-->

**<div id="nav-bar">**

      <span>home &#9660;</span>

      <span>categories &#9660;</span>

      <span>deals &#9660;</span>

      <span>sale &#9660;</span>

      <span>profile &#9660;</span>

    </div>

    <!--Visible page content of the web store-->

    <div id="content">

      <p id="task">Scroll down!</p>

      <div id="placeholder">

        <!--This div's purpose is to create a content long to make the site scrollable-->

      </div>

    </div>

</body>

</html>

<!DOCTYPE html>

#### Position: relative (to itself)

If you assign the CSS property position: relative, the element will (still) sit at its original position. However, using the **four location properties** you can now shift the HTML element **relative** to its normal position.

Take a look at the next example where the sky and space switched their original order and where the landscape even escaped the universe…

<html>

<head>

  <title>**Position Relative**</title>

  <style>

      \* {

        text-transform: uppercase;

        height: 40px;

        font-family: sans-serif;

        line-height: 40px;

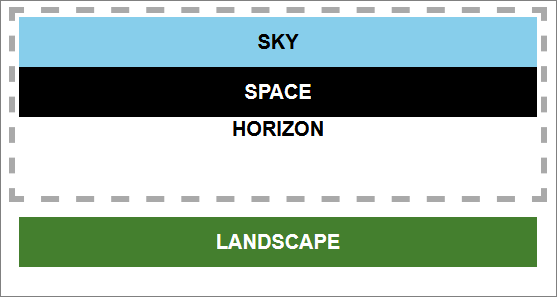
        text-align: center;

        font-weight: bold;

      }

      #universe {

        height: initial;

        padding: 3px;

        border: 5px dashed darkgrey;

      }

      #space {

        color: white;

        background-color: black;

        position: relative;

        top: 40px;

      }

      #sky {

        background-color: skyblue;

        position: relative;

        bottom: 40px;

      }

      #horizon {

        height: 20px;

        line-height: 20px;

* The landscape element’s position is set to relative and the location property top is set to **60px**. This results in a position 60px below the original position.
* The space and sky elements have a **relative position**, too. Their position is changed by the location properties top and bottom, respectively, each with a value of 40px (the height of both elements).

      }

      #landscape {

        color: white;

        background-color: #447F2E;

        position: relative;

        top: 60px;

      }

</style>

</head>

<body>

    <div id="universe">

        <div id="space">space</div>

        <div id="sky">sky</div>

        <div id="horizon">horizon</div>

        <div id="landscape">landscape</div>

      </div>

</body>

</html>

<!DOCTYPE html>

#### Position: absolute

When assigning the CSS property position: absolute to an element, **the parent element and other ancestors become relevant**, since the HTML element will be positioned **relative to its nearest ancestor** with an **explicit position**.

With **explicitly positioned ancestors** we mean all HTML elements which contain the styled element as a child in any level and have either a fixed, relative or absolute position.

<html>

<head>

  <title>**Position Absolute**</title>

  <style>

      #great-grandmother {

**position: absolute;**

**top: 10%;**  /\* from <body> \*/

**left: 10%;**

      }

      #grandmother {

**position: absolute;**

**top: 10%;**  /\* from great-grandmother \*/

**left: 10%;**

      }

      #mother {

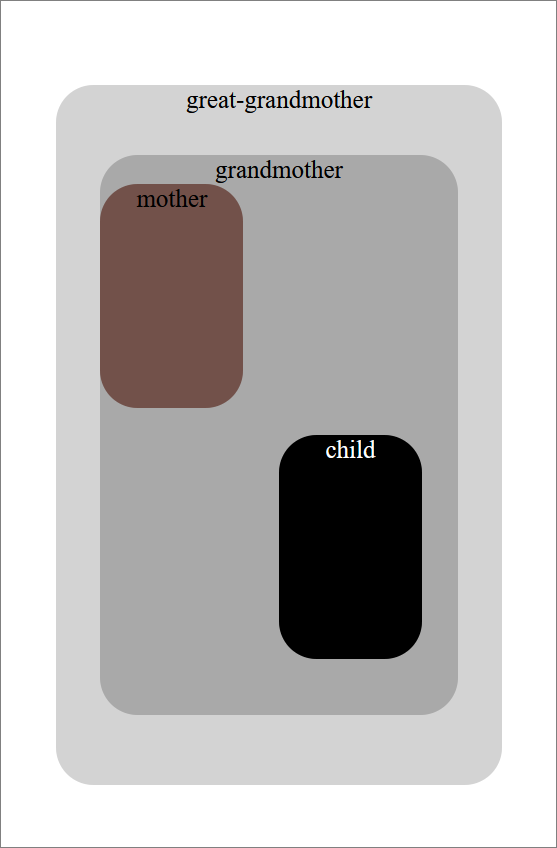
        /\* **not positioned**, so it's placed right after the grandmother \*/

      }

      #child {

**position: absolute;**

**bottom: 10%;** /\* from grandmother as mother is NOT positioned \*/

        **right: 10%;**

      }

      \* {

        text-align: center;

        font-size: 20px;

        border-radius: 30px;

      }

      #great-grandmother {

        height: 80%;

        width: 80%;

        background-color: lightgrey;

      }

      #grandmother {

        height: 80%;

        width: 80%;

        background-color: darkgrey;

      }

      #mother {

        height: 40%;

        width: 40%;

        background-color: #72514A;

      }

      #child {

        height: 40%;

        width: 40%;

        color: white;

        background-color: black;

      }

  </style>

* The great-grandmother is positioned absolute and therefore is positioned **relative to the body** of the HTML document.
* It’s the same with the **grandmother**, except that she is not positioned relative to the body, but to the **great-grandmother** as she is the first ancestor with an **explicit position**.
* The **mother** doesn’t have the **position** property and therefore is simply placed directly after the content of her parent.
* The **child** has an **absolute position** again. The child doesn’t get positioned relative to the mother, because the mother doesn’t have an **explicit position**. The next ancestor with an **explicit position** is the grandmother and therefore the child is positioned relative to her.

</head>

<body>

    <div id="great-grandmother">great-grandmother

      <div id="grandmother">grandmother

        <div id="mother">mother

          <div id="child">child

          </div>

        </div>

      </div>

    </div>

</body>

</html>