Key concept: position property

Key concept: position property

The 'left', 'top', 'right', and 'bottom' properties

In CSS, there are four properties which can be used to adjust or set the position of an element. Those properties are: `left`, `top`, `right`, and `bottom`.

However, in one of the best jokes played by the authors of the CSS specification, using those properties by themselves will have *no effect* on any element. Surprisingly, most developers struggling with CSS don't find this funny.

The reason these properties don't work by default is that they only work when applied to *positioned* elements. And positioned elements are those that have had their `position` property changed from the default.

The 'position' property

The CSS `position` property governs how an element is positioned on the page and how it responds to the position adjusting properties (`left`, `top`, `right`, and `bottom`). Up to this point in this course, we have not used this property and so everything we've seen has been the default position, which is `position:static` for all elements.

As of today, the position property has four different values it can take: `static`, `relative`, `absolute`, and `fixed`. We will look at `static` and `fixed`. The options `relative` and `absolute` are more complex, they'll be discussed in an optional advanced section for completeness, but we aren't going to worry much about them because flexbox reduces their use cases.

Static

```
position: static; /* the default */
```

A position property of static is the default for all elements. It simply means that all elements follow the "flowing text"model of layout and the only properties influencing their position are margins, padding, and the display property that selects block level layout, inline or inline-block. Static elements *ignore* the positioning properties we read about earlier (`left`, `top`, `right`, and `bottom`).

Fixed

```
position: fixed;
```

A fixed positioned element respects the positioning properties (`left`, `top`, `right`, and `bottom`). A fixed positioned element is positioned against the *window* rectangle (aka the viewport). This means that fixed position elements will not scroll with the rest of the page. Fixed position elements can easily "stick" to the side or bottom or top of the browser. To observe this better, see the Fixed position activity in the next section.

- Fixed position elements are positioned against the viewport.
- Best practice: use both a horizontal and a vertical positioning property on every fixed positioned element.
- Fixed positioned elements do not contribute to size of the parent.
- Fixed positioned block level elements do not get the width of their parent.
- Margins do not work the same.
- Opposite properties can be used to size an element.

Best practice: use both horizontal and vertical positioning property on every fixed element

There is a very subtle extension to the previous interpretation problem: if an element is set to be `position:fixed`, but has no horizontal positioning property (that is, `left` or `right`), then it will be displayed in the flow exactly as it would have been. Except, later, if `left:0px`; is added (for example), then the element may jump to the left edge of the browser window. The same applies vertically. This is a bewildering behavior, as most users do not expect there to be a difference between `left:0px` and no `left` property at all.

Therefore, for any fixed positioned element, the best practice is to ensure that one of the horizontal positioning properties (that is, `left` or `right`) and one of the vertical properties (`top` or `bottom`) are both set.

Fixed positioned block level elements do not get the width of their parent

Earlier we learned that the block level element automatically gets the width of its parent, that is, they extend to become full width. But this is only true for `static` and `relative` positioned elements. Elements that are `fixed` positioned (or `absolute`) do not exhibit this behavior. Their initial width is simply the width of their content. Though it can be changed.

Margins do not work the same

For `static` and `relative` positioned items, margins can be used to adjust an element position and keep neighboring siblings "away". That's our quick assumption about the margins. However, when an element is `fixed` positioned, a given margin *might* be able to move the element but will not move any siblings. Margins cannot be used to keep siblings "away", to fight crowding.

As a general rule, if a positioning property is being used (like `left`), then the matching margin (`margin-left`) can also be used to additionally adjust the position of the element. Otherwise, the margin will unlikely have any effect.

opposite properties can be used to size element

This is one of the nicer features. Working with preset dimension properties (`height` and `width`) can make your design brittle and reduce its adaptability. However, `fixed` positioned items can instead set the opposite positioning properties (like `left` and `right`) and leave the matching dimensional property (`width`) unspecified. The element will grow or shrink based on the size of the browser window. Note that this feature is only available to `fixed` (and `absolute`) elements.