ISTA 230

Positioning

CSS

Position

By default, elements are laid out on the page in the order in which they're specified within the HTML.

Block elements are laid out vertically on top of one another, taking up as much room as they need (as specified by the CSS properties of width, height, padding, border, and margin).

Inline elements are laid out horizontally one after another in the order in which they're specified. Inline elements do not have explicit dimensions but are sized based on the content within them. Inline elements can have horizontal margins/padding but vertical margin/padding values have no effect on them. If necessary, inline elements will be displayed across multiple lines.

The position of elements as described above is known as 'static' positioning.

CSS

Position

Using the CSS 'position' property, we can adjust how elements are laid out on the page. By default, a paragraph with a span child would look like this:

Browser

This is a small chunk of content. Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

By using 'relative' for our position property, we can tell the browser that our span should be positioned 'relative' to its default location. We'll specify 'top' and 'left' values for the element to specify vertical and horizontal offset from the initial position of the element.

HTML

```
<p>This is a <span>small chunk</span> of content...</p>
```

CSS

```
span
{
   position:relative;
   top:30px;
   left:30px;
}
```

Browser

This is a of content. Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal. small chunk

Note that the top and left properties have 'right' and 'bottom' counterparts. If a 'top' value is specified for an element that is relatively positioned, then the 'bottom' value will be ignored. Similarly, if a relatively positioned element has a 'left' property value, then the 'right' property will be ignored.

CSS

Absolute Position

We can also specify that an element should have an 'absolute' position.

- When using 'position:absolute', the following things happen:
    - The element is taken out of the document flow
    - Position is determined based on the containing element
    - Position **and dimensions** are set using 'top','bottom','left', and 'right'

Absolutely positioned elements are displayed based on their "containing element". The containing element is **NOT** the same as a parent element.

The containing element is the nearest ancestor with a non-static 'position' value. If no ancestor with a non-static position value is found, then the position is based on the topmost element (i.e., the <html> element).

Unlike relatively positioned elements, absolutely positioned elements can have 'top', 'bottom', 'left', and 'right' properties specified and all will be applied. If both 'top' and 'bottom' properties are specified, they will be used to calculate both the position and the height of the element (i.e., the height of the element will be the difference between the top and bottom properties). Similarly, if 'left' and 'right' are both specified, they will be used to calculate both the position and the width of the element (i.e., the width of the element will be the difference between the left and right properties).

When an element is positioned absolutely, it is removed from the normal document flow. That means that if all elements within a parent element are positioned absolutely, the element will collapse on itself (similarly to what was discussed with 'floated' elements). When this happens, you will not be able to 'uncollapse' the parent element using 'overflow: auto'. Instead, you'll need to explicitly set the height of the parent element.

Lastly, when an element is positioned absolutely, all CSS properties that can be specified using relative units (e.g., 50%) will be based on the **containing element**, not the parent element.

CSS

Fixed Position

We can also specified that elements should have a fixed position. This is similar to absolute positioning:

- The element is taken out of the document flow
- Position **and dimensions** are set using 'top','bottom','left', and 'right'
- **BUT, the viewport (i.e., browser window) is always the containing element**

CSS

position Property

CSS

```
div { position: relative; }
```

```
div { position: absolute; }
```

```
div { position: fixed; }
```

```
/*
 * Position is used to specify how an element
 * should be positioned on the page. When using
 * 'absolute' positioning, remember the rules
 * about containing elements. For 'fixed',
 * remember that the viewport is the containing
 * element.
 *
 *
 * Default Value: static
 * Inherited: No
 *
 */
```

CSS

Natural Stacking Order

We usually think about our websites in terms of the x-axis and the y-axis. Elements are laid out horizontally along the x-axis and vertically along the y-axis. These are the contexts for much of what we've discussed thus far in the semester. However, there is a third axis, the z-axis, that comes into play when we discuss how things are stacked on top of one another as they are laid out on the page.

It might be helpful to think of the elements on the page as a deck of cards. As each element is drawn on the screen, it is 'stacked' on the top of the deck. The next element is then added to the stack, followed by the next, and so on until the entire page is rendered. Because most of our elements don't occupy the same space on the screen, the concept of stacking isn't something that users (or designers) intuitively notice.

Elements are stacked on the webpage in the following order:

- Background/border of main element (e.g. <body>)
- **Positioned elements with a negative stacking context**
- Block-level, non-positioned, non-floated elements
- Non-positioned, **floated** elements
- Inline elements
- **Positioned elements**

When an element is given a non-static position value, it's stacking order can be adjusted using the z-index property.

It is worth noting that when an element is given a non-'auto' value for z-index, a new stacking context is created. Stacking contexts are one of the more complex concepts in web design so an analogy might be in order to help us understand.

Imagine you're sorting all of the paperwork for your accountant to do your taxes. All of your documents are going to be placed inside a large envelop. To make things easier for your accountant, you decide that you'll organize your papers by type into smaller envelopes and then place them all inside of the larger envelope.

In the analogy above, the larger envelope represents the stacking context created by the <body> element. The smaller envelopes represent the new stacking contexts created by your non-'auto' z-index elements. For the descendents of these elements, any z-index value is relative to other elements within that stacking context. However, it does not take into consideration any elements outside of that stacking context.

CSS

z-index Property

CSS

```
    div { z-index: 10; }
```

```
    div { z-index: -10; }
```

```
    /*
     * Allows us to change the stacking order of
     * elements.
     * This property only applies to non-static
     * positioned elements!
     *
     *
     * Default Value: auto
     * Inherited: No
     *
     */
```