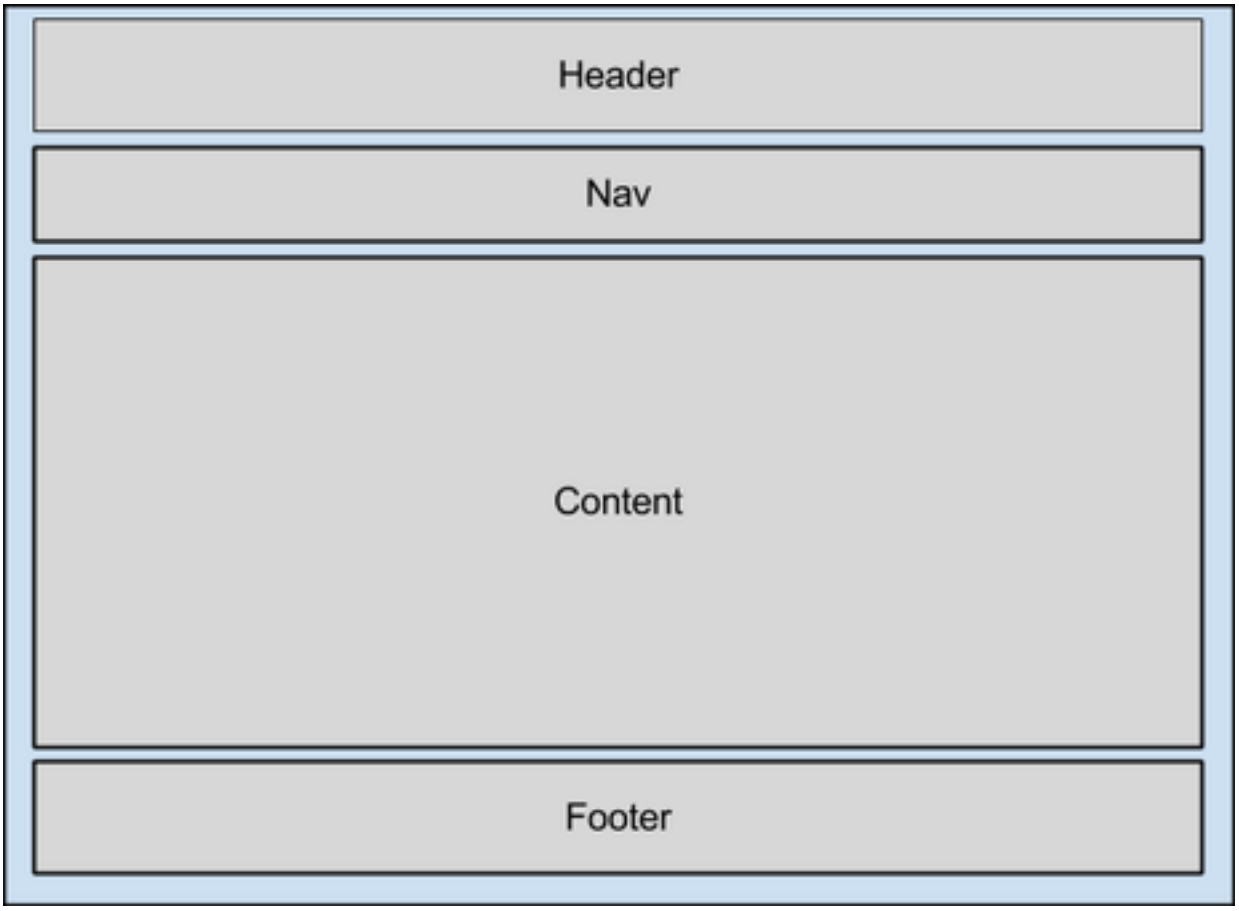
Position

Document flow

To understand how to position we need to first understand the standard flow of a document. When HTML is written out it the elements are displayed in the order they are written in the code. This is known as the “normal flow” of the document. While the flow is standard and might make sense it can be limiting because the elements must display in the order the are written.



Position

Property Values

Value	Description
static	Elements renders in order, as they appear in the document flow. This is default.
absolute	The element is positioned relative to its first positioned (not static) ancestor element
fixed	The element is positioned relative to the browser window
relative	The element is positioned relative to its normal position, so "left:20" adds 20 pixels to the element's LEFT position
inherit	The value of the position property is inherited from the parent element

Positioning Properties

Property	Description
bottom	Specifies the bottom position of a positioned element
left	Specifies the left position of a positioned element
right	Specifies the right position of a positioned element
top	Specifies the top position of a positioned element

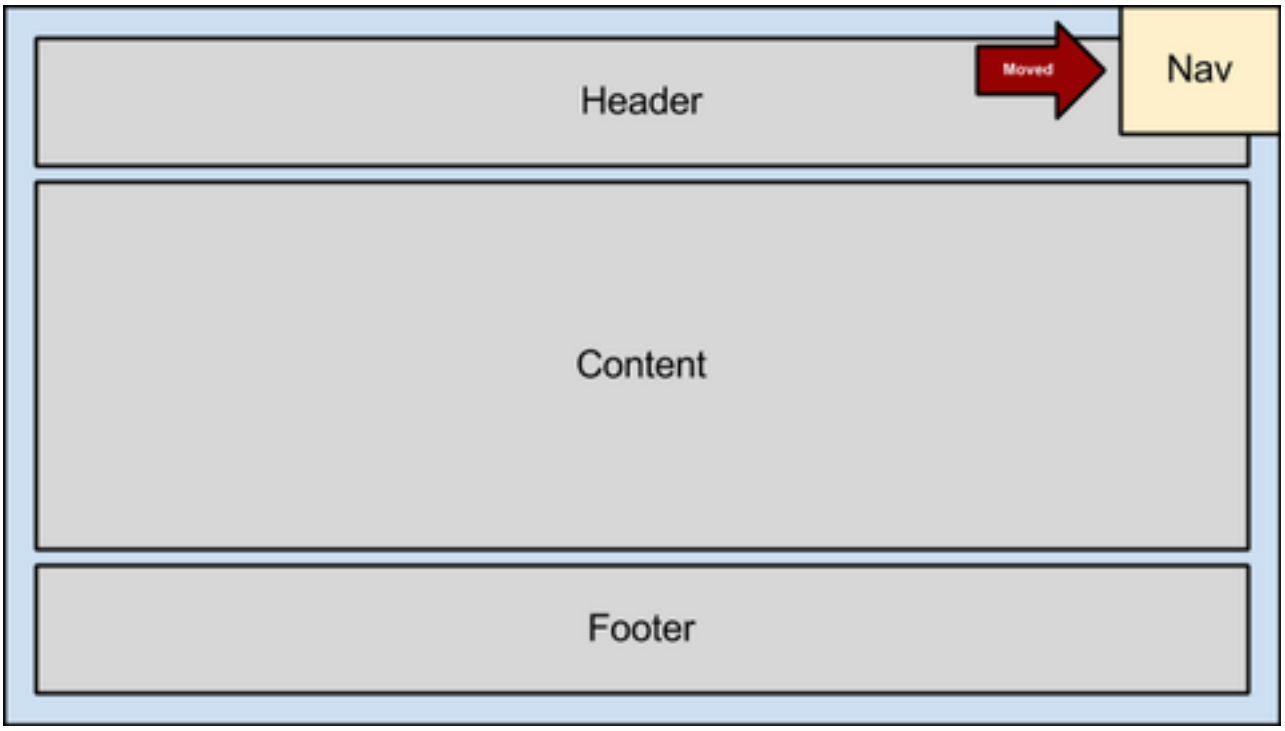
Static

position: static is the normal or default style for all elements. Since this is implied on all elements the only time it needs to be set is when an element needs to override a style and set it back to static.

.sva{css} Position

Absolute

position: absolute will pull the element out of the “normal flow” and layer it on top of the document. That element than can be moved to a position on the page with the addition of TOP, RIGHT, BOTTOM or LEFT properties.



The nav element was pulled out of the flow of the document and positioned in the top right corner of the document. Since the nav was pulled out of the normal flow of the document its space is not saved and the elements below move up and fill in the available space.

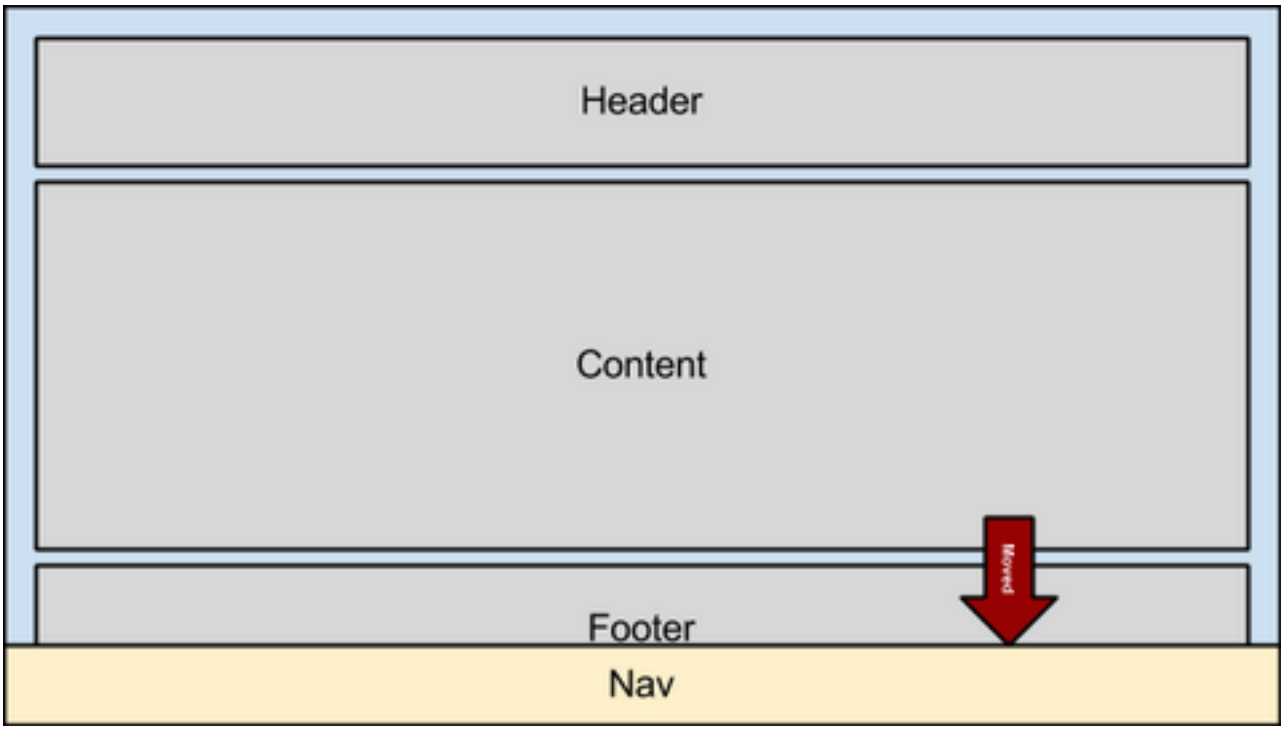
CSS CODE

```
#nav {
        position: absolute;
        top: 0;
        right: 0;
}
```

.sva{css} Position

Fixed

position: fixed is similar to absolute. Both pull the element out of the flow of the document and moves to the location set by top, right, bottom or left. But with fixed the element is locked to the browser window and not the document content. When and element is fixed and the browser window scrolls that element will stay in the same location.



Notice the nav goes across the bottom of the window. Since we can't put a width of 100% on the element if it has padding because of the box model we need to do this in a different way. If we put both a left and right value the element will stick to both sides of the window. This will make the element fill the whole space.

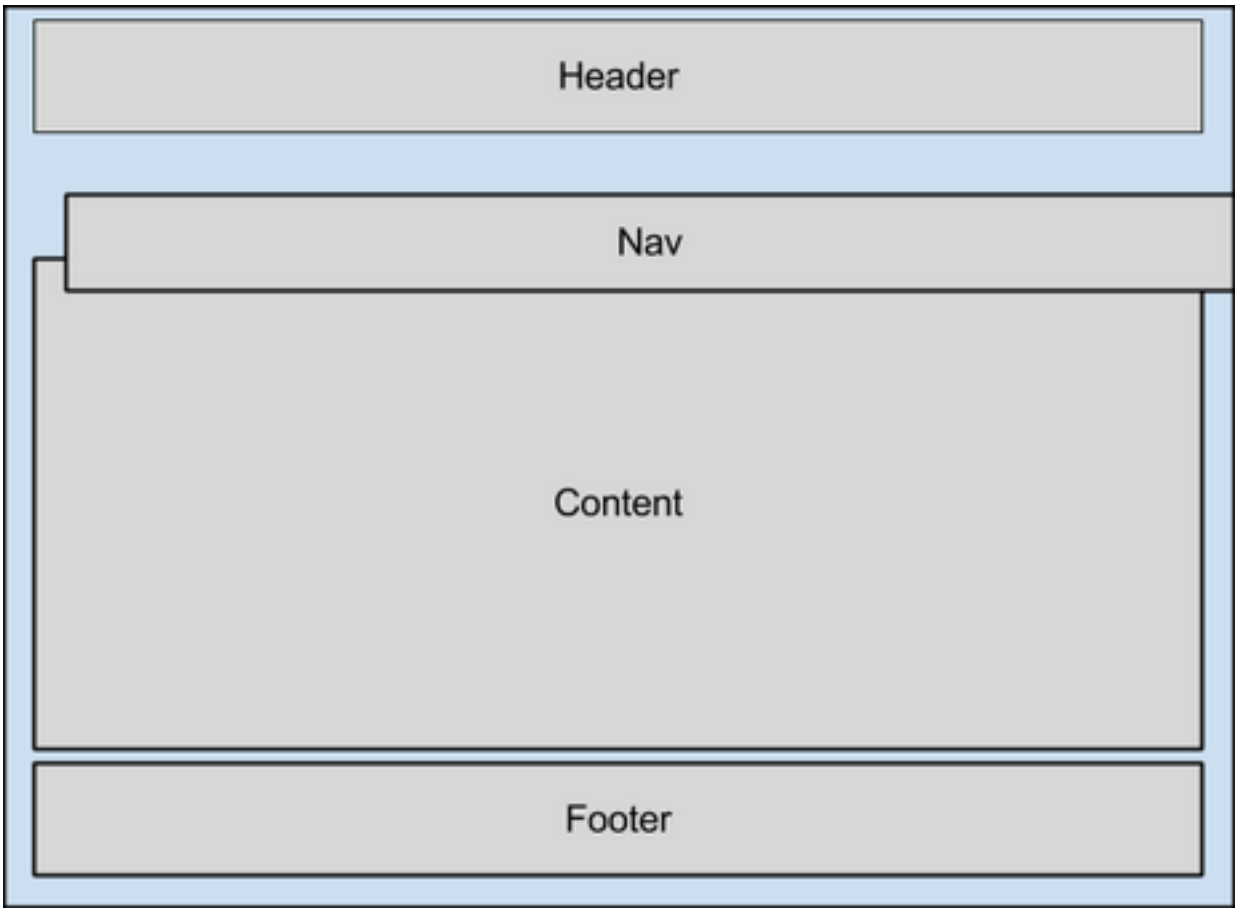
CSS CODE

```
#nav {
        position: fixed;
        bottom: 0;
        left: 0;
        right: 0;
}
```

.sva{css} Position

Relative

position: relative will move the element when a top, right bottom or left is applied but unlike absolute the element will not be removed from the normal flow of the document. Since the element is still in the flow of the document a space will be kept for it and you may see some white space around the element.



Notice how the nav element moves but the space is left where it was.

CSS CODE

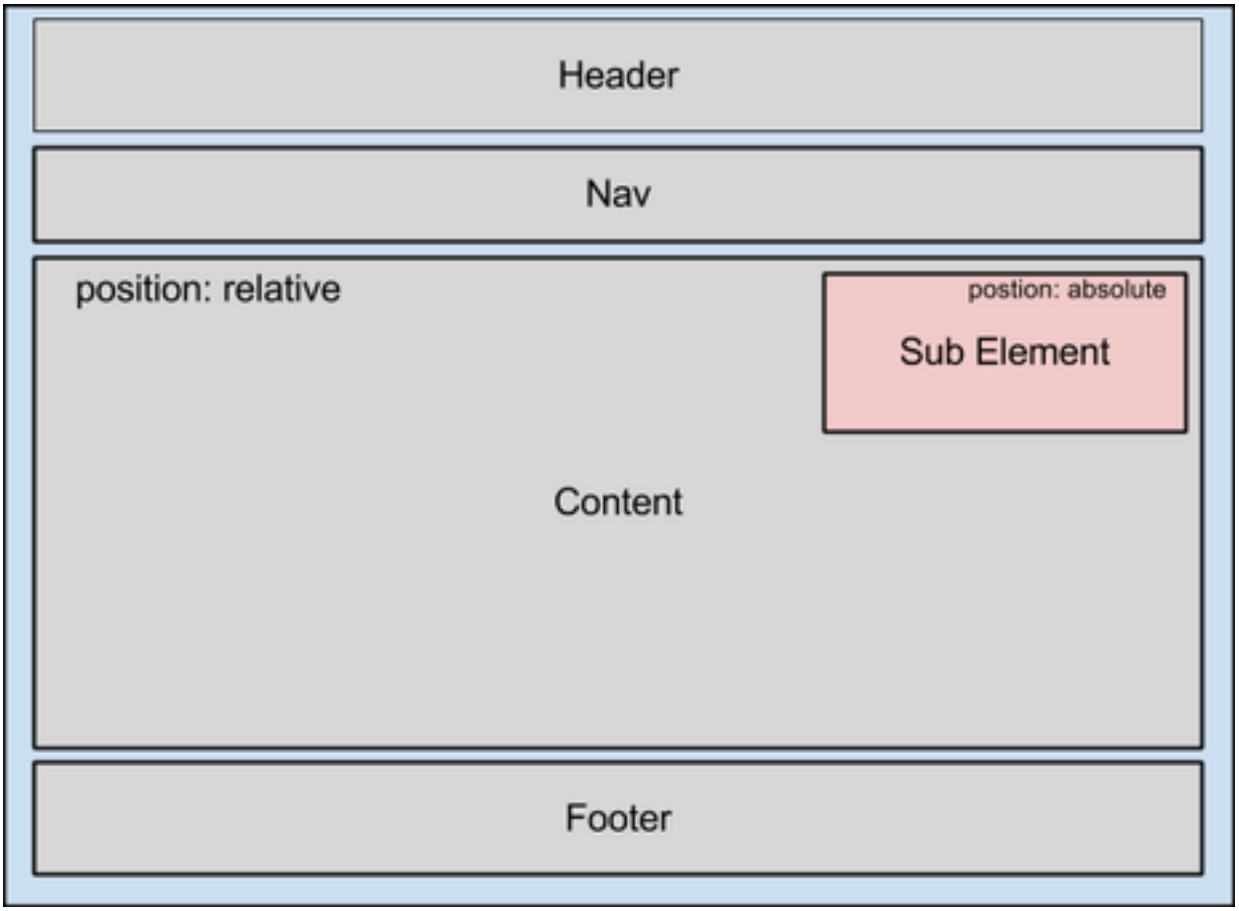
```
#nav {
        position: relative;
        top: 10px;
        right: 10px;
}
```

While this may not seem like it is that useful of a property it has a very valuable side effect.

Position

Relative + Absolute

When relative and absolute work together something magical happens. When an element has a position absolute it will go to the extremes of the document. This is not always the most useful behavior but if its parent element has a position of relative it is like creating a new document with in the main document. When any child element has a position absolute it will define its position based on the parent element with relative position rather than the document.



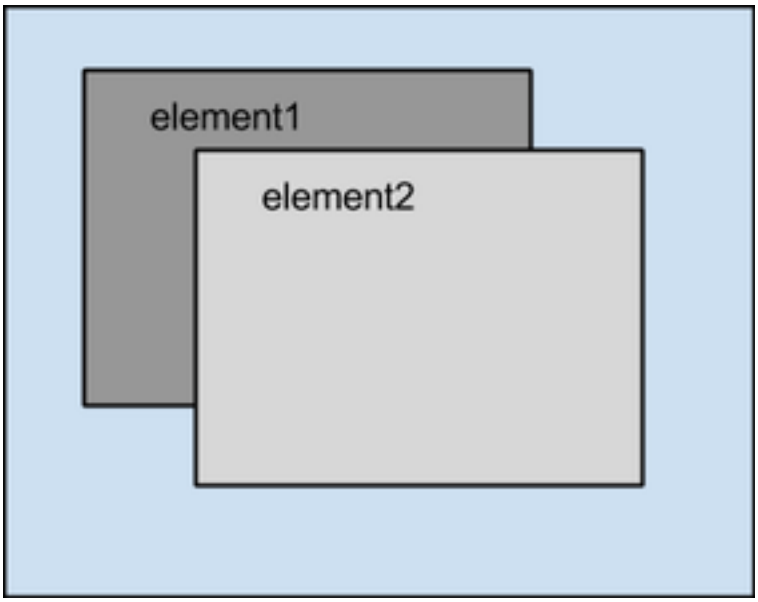
Position

CSS CODE

```
#content {
        position: relative;
}
#nav {
        position: absolute;
        top: 10px;
        right: 10px;
}
```

Stacking Elements

There may be times when two or more elements may take up the same space when they are positioned absolutely. When this happens the elements will stack on top of each other. The order depends on the “flow order”, meaning the first element written to the page will be the lowest. The later the element written the higher it is in the stack.



Z-index controls the stacking order of an element. All elements have a default z-index of 1, since they are the same the flow is the tiebreaker and the first element is lower.

To get around this and have the first or lower element stack on top of the other elements you need to add a z-index. In this example element1 will stack on top of element2 because it has a higher z-index.

Position

```
.element1 {
        position: absolute;
        top: 10px;
        left: 10px;
        z-index: 5;
}
.element2 {
        position: absolute;
        top: 10px;
        left: 10px;
        z-index: 4;
}
```

Visibility

Visibility is like setting **display: none** on an element with one major difference. With visibility set to **hidden** the element will not appear but the space in the document where the element was is maintained. With display: none; the element is removed from the flow of the document and the space will collapse.

Property Values

Value	Description
visible	The element is visible. This is default
hidden	The element is invisible (but still takes up space)
collapse	Only for table elements. collapse removes a row or column, but it does not affect the table layout. The space taken up by the row or column will be available for other content. If collapse is used on other elements, it renders as "hidden"
inherit	Specifies that the value of the visibility property should be inherited from the parent element

Position