LAYOUT-
- determining what type of layout you want your site to have will dictate how you position elements.
- Questions to ask yourself:
  - Fixed-Width? or Elastic?
  - Centered Layout? or Left Aligned?
  - How many columns?
  - Where will Navigation go?
  - Where will Content go?
Fixed Width & Centered Layout

body {
    padding: 0;
    margin: 0;
    text-align: center;
}

#wrapper {
    width: 800px;
    margin-left: auto;
    margin-right: auto;
    text-align: left;
}

-includes bug fix for IE

-the wrapper is your “live” area centered in the browser window
Absolute Positioning

With absolute positioning, you can precisely locate elements on your page, potentially irrespective of where they happen in the code. With CSS positioning, you need to be cognizant of the “normal flow” of the document, since the positioning values relate to that.

```css
position: static;
position: relative;
position: absolute;
position: fixed;
```
position: static;
this is the default, and means that you aren’t positioning the element. It will display in the normal flow of the document. There are not very many occasions to specify this.

position: relative;
this positions an element relative to where it would have been in the normal flow of the document - perhaps to the right or left of where it normally would have been. Relative positioning is useful not so much as a way to position something, but instead as a way of creating a new point of reference for an absolutely positioned element nested within it.
position: absolute;

this positions the element relative to the body tag and removes it from the normal flow of the document.

- Use the left or right and top or bottom properties to locate that element “exactly.”

  top:-----; left:-----;

- The unit of measurement can be fixed (pixels) or relative (ems, percentage).

Note: if you nest an element with absolute positioning within a element that has absolute, relative or fixed positioning already, the positioning is no longer measured from the edges of the browser window, but from the edges of the containing element that has already been positioned absolutely.
```css
position: fixed;
```

this positions the element specifically within the browser window and “fixes” it (as if it were a sticky note stuck to the monitor). Things will scroll beneath it, but the element will remain in the same location. This is used to create a non-scrolling element like a navigation bar. Fixed positioning is always relative to the screen, regardless of whether you have nested that element within another positioned element.

STICKY NOTE

When positioning an element, specifying the location of two edges is usually enough. The values you give for positioning measure from the edge of the browser window. You can specify both left and right position, but IE is buggy with this, so it is best instead to specify a width, rather than a left and right edge position. IE is also buggy with bottom and right positioning.
You can specify the heights of elements on the page, but unless the element has itself a fixed height, you don’t necessarily know the height of what will display in the specified region (text might display bigger, for instance, and spill over the edges of the region). The overflow property can help with that situation.

**overflow: auto**
auto will add scrollbars where necessary

**overflow: hidden**
hidden will prevent content that is too large to display. Can be useful as a sort of work-around for some kinds IE bug solutions.

**overflow: visible**
this is the default, and will automatically display your content (even if larger than its container).

**overflow: scroll**
this gives you scrollbars automatically (even if you don’t need them). Auto is a good choice when you only want scrollbars when necessary (say, for vertical, not horizontal, scrolling).
Creating a customized underline

```css
a {
  text-decoration: none;
  background-image: url(images/underline.gif);
  background-repeat: repeat-x;
  background-position: left bottom;
}
```

The line has to repeat itself to fit for different sized links, so don’t do anything that can’t be repeated seamlessly (flourishes, etc).

Creating a “button”

```css
a {
  color: #ffffff;
  background-color: #339900;
  font-weight: bold;
  border-width: 4px;
  border-style: solid;
  border-top-color: #999966;
  border-left-color: #999900;
  border-right-color: #663333;
  border-bottom-color: #663333;
  padding: 2px;
}
```

This gives a “beveled button” appearance by using different colored borders to create the illusion of dimension. The anchor tag is an inline element, so margins on top and bottom don’t work.
Using graphics with links

```javascript
a {
    background-image: url(images/bullet.gif);
    background-repeat: no-repeat;
    text-decoration: none;
    background-position: left center;
    padding-left: 30px;
    padding-top: 10px;
    padding-bottom: 10px;
}
```

Loads an image in the background, positioned to the left of the text. The padding on the left allows the text to move to the right and not overlap the graphic. The padding on top and bottom just allows the image to display but only the size of the graphic; doesn’t add more.

If you want more, you need to convert the anchor tag from an inline to block element:
```
display: block
```