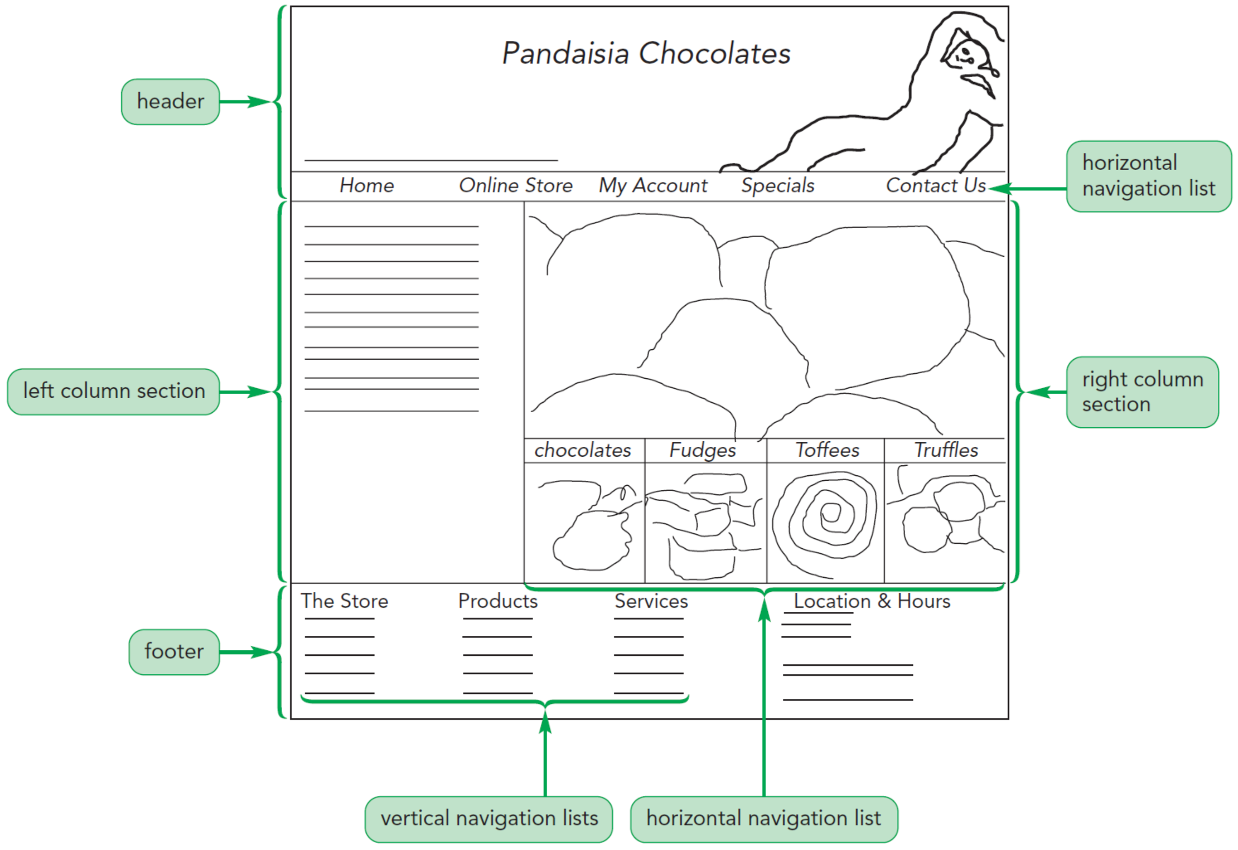
Course Notes – CSS 2: Web Design and Simple Layout

Page Layout Designs

Start from a sketch

Example:

Figure 3–4 **Proposed home page layout**



Display Style

Figure 3–1 **Some values of the display property**

Display Value	Appearance
`block`	Displayed as a block
`table`	Displayed as a web table
`inline`	Displayed inline within a block
`inline-block`	Treated as a block placed inline within another block
`run-in`	Displayed as a block unless its next sibling is also a block, in which case, it is displayed inline, essentially combining the two blocks into one
`inherit`	Inherits the display property of the parent element
`list-item`	Displayed as a list item along with a bullet marker
`none`	Prevented from displaying, removing it from the rendered page

Start Clean State

Example

```css
article, aside, figcaption, figure,
footer, header, main, nav, section {
   display: block;
}

/* Typographic Styles */

address, article, aside, blockquote, body, cite,
div, dl, dt, dd, em, figcaption, figure, footer,
h1, h2, h3, h4, h5, h6, header, html, img,
li, main, nav, ol, p, section, span, ul {
   background: transparent;
   font-size: 100%;
   margin: 0;
   padding: 0;
   vertical-align: baseline;
   box-sizing: border-box;
}

nav ul {
   list-style: none;
   list-style-image: none;
}

nav a {
   text-decoration: none;
}

body {
   line-height: 1;
}
```

Understanding the CSS position Property

The CSS position property provides developers with the ability to control the placement of elements on a web page. It has several values, each with its specific use case and behavior. Here are the four primary values for the position property:

1. static

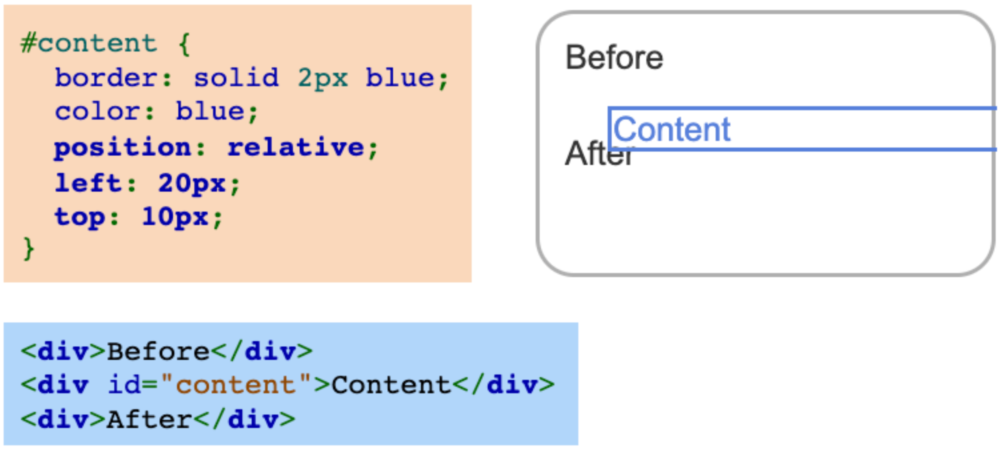
- **Description**: Static positioning is the default value for the position property. Elements with position: static are positioned according to the normal flow of the document. They are not affected by the top, bottom, left, and right properties.
- **Use Case**: Default positioning for most elements where no special positioning is required.

```
.static-element {
    position: static;
  }
```

2. relative

- **Description**: Relative positioning positions the element relative to its default position. The element will still occupy the same space in the document flow, but it can be moved using the top, bottom, left, and right properties.
- **Use Case**: Adjusting the position of an element slightly from its normal position without affecting the layout of surrounding elements.

```
.relative-element {
    position: relative;
    top: 10px;
    left: 20px;
  }
```



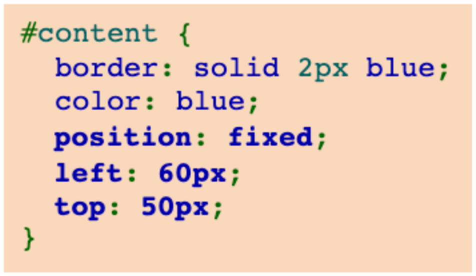
```
#content {
  border: solid 2px blue;
  color: blue;
  position: relative;
  left: 20px;
  top: 10px;
}
```

```
<div>Before</div>
<div id="content">Content</div>
<div>After</div>
```

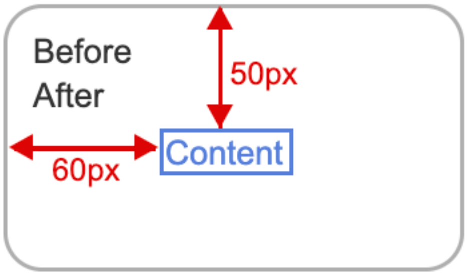
3. fixed

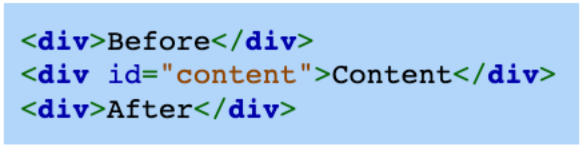
- **Description**: Fixed positioning positions the element relative to the viewport, meaning it stays in the same place even when the page is scrolled. Elements with position: fixed are removed from the normal document flow.
- **Use Case**: Creating elements that should remain visible at all times, such as a sticky header or a floating navigation bar.

```
.fixed-element {
    position: fixed;
    top: 0;
    right: 0;
  }
```



```
#content {
  border: solid 2px blue;
  color: blue;
  position: fixed;
  left: 60px;
  top: 50px;
}
```





```
<div>Before</div>
<div id="content">Content</div>
<div>After</div>
```

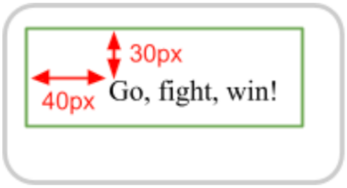
4. absolute

- **Description**: Absolute positioning positions the element relative to the nearest positioned ancestor (an ancestor with a position value other than static). If no such ancestor exists, it positions the element relative to the initial containing block (usually the document body). Elements with position: absolute are removed from the normal document flow.
- **Use Case**: Placing elements in an exact position relative to a container or another element.

```
.absolute-element {
    position: absolute;
    top: 50px;
    left: 100px;
  }
```

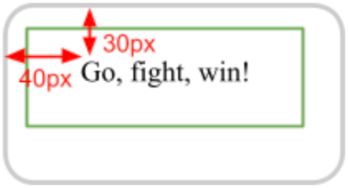
```
#container {
  border: solid 2px green;
  position: relative;
  height: 60px;
  width: 150px;
}

#cheer {
  position: absolute;
  left: 40px;
  top: 30px;
}
```

```
<div id="container">
  <div id="cheer">Go, fight, win!</div>
</div>
```



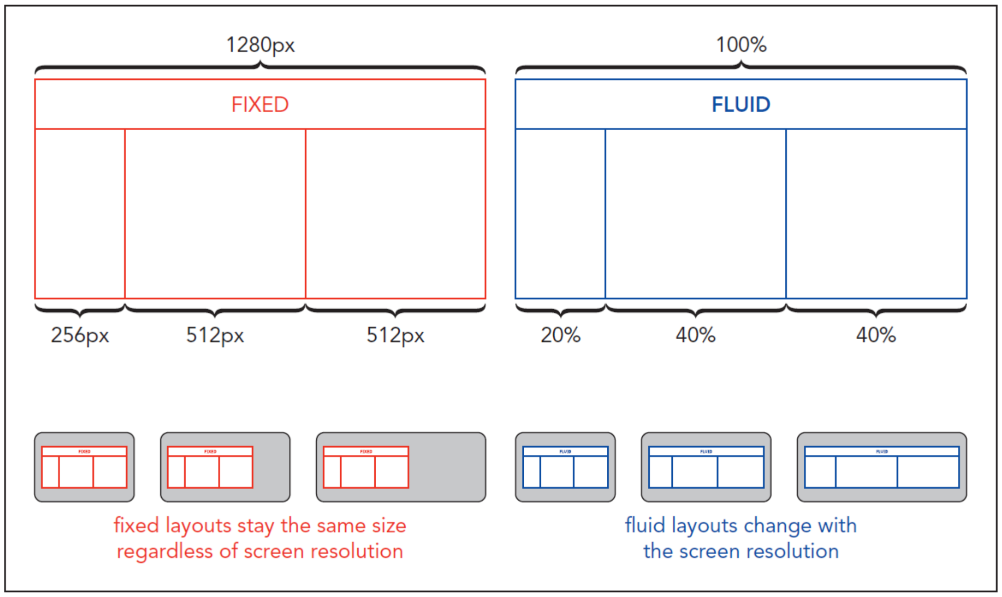
```
#container {
  border: solid 2px green;
  /* No positioning */
  height: 60px;
  width: 150px;
}

#cheer {
  position: absolute;
  left: 40px;
  top: 30px;
}
```

```
<div id="container">
  <div id="cheer">Go, fight, win!</div>
</div>
```



Fixed vs Liquid Layouts

Figure 3–5 **Fixed layouts vs. fluid layouts**


1280px
FIXED
256px
512px
512px
100%
FLUID
20%
40%
40%
fixed layouts stay the same size regardless of screen resolution
fluid layouts change with the screen resolution

There is also Responsive Design that is design that changes according to the display size and type.

Width and Height

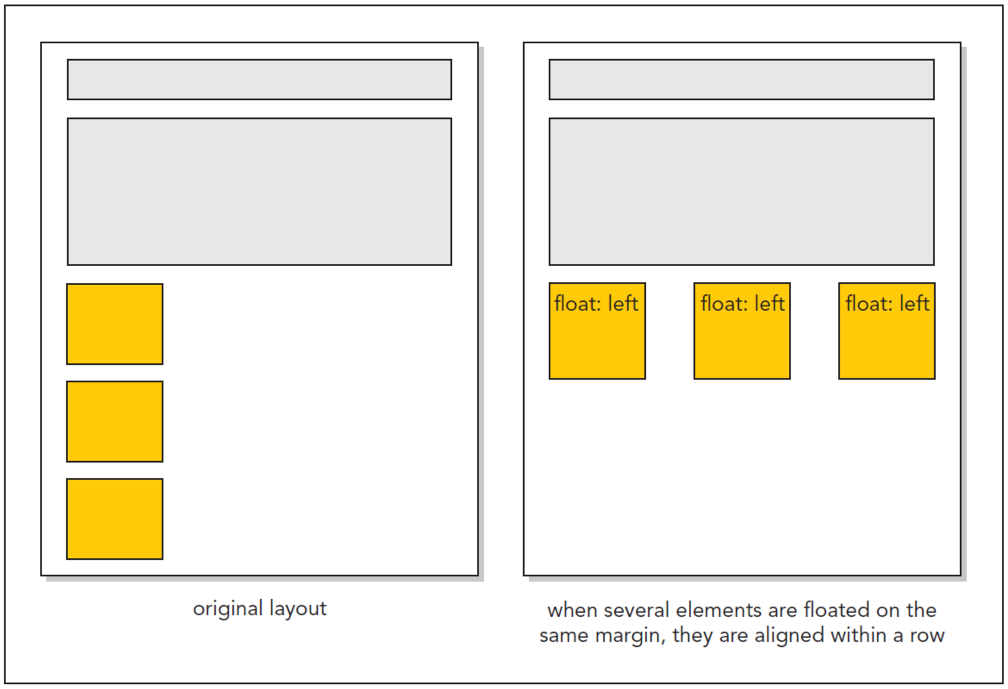
```
body {
  width: 95%;
  min-width: 640px;
  max-width: 1680px;
}

body > header > img {
  display: block;
  width: 100%;
}
```

Centering a Block Element

```
body {
  margin-left: auto;
  margin-right: auto;
  width: 95%;
  min-width: 640px;
  max-width: 1680px;
}
```

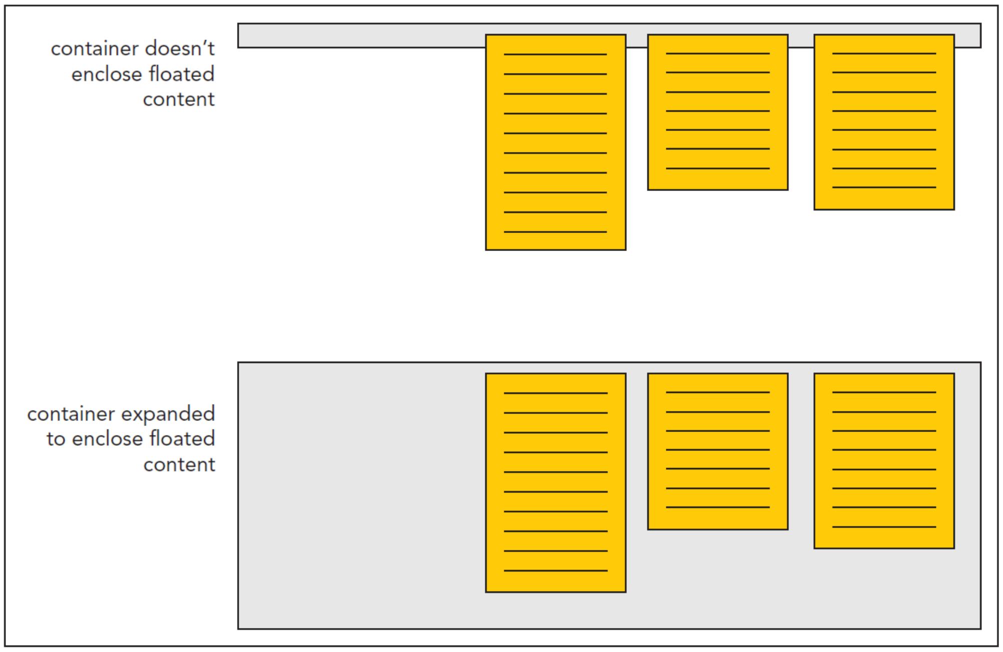
Float

Figure 3–10 **Floating multiple elements in a row**


float: left
float: left
float: left
original layout
when several elements are floated on the same margin, they are aligned within a row

Container Collapse

Figure 3–26 **Container collapse**


container doesn't enclose floated content
container expanded to enclose floated content

```
container::after {
  clear: both;
  content: '';
  display: table;
}
```

See: Layout Folder