CSS LAYOUT

The position Property

- The position property specifies the type of positioning method used for an element.
- There are four different position values:
    1. static
    2. relative
    3. fixed
    4. absolute
    5. sticky

Static

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.
- An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

```
div.static {
    position: static;
    border: 3px solid #73AD21;
}
```

relative

- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

```
div.relative {
    position: relative;
    left: 30px;
    border: 3px solid #73AD21;
}
```

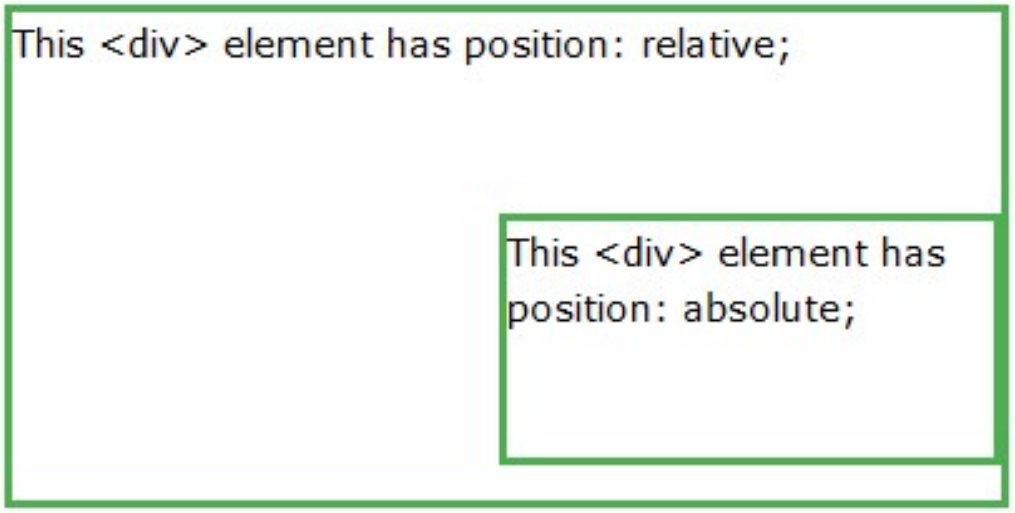
fixed

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

```
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;
}
```

absolute

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.



sticky

- An element with position: sticky; is positioned based on the user's scroll position.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

```
div.sticky {
    position: -webkit-sticky;
    position: sticky;
    top: 0;
    background-color: green;
    border: 2px solid #4CAF50;
}
```

Overlapping Elements

- The z-index property specifies the stack order of an element.
- An element with greater stack order is always in front of an element with a lower stack order.

CSS Layout - inline-block

- It has been possible for a long time to create a grid of boxes that fills the browser width and wraps nicely (when the browser is resized), by using the float property.
- However, the inline-block value of the display property makes this even easier.
- inline-block elements are like inline elements but they can have a width and a height.

The old way - using float

```
.floating-box {
    float: left;
    width: 150px;
    height: 75px;
    margin: 10px;
    border: 3px solid #73AD21;
}

.after-box {
    clear: left;
}
```

The same effect can be achieved by using the inline-block value of the display property (notice that no clear property is needed):

```
.floating-box {
    display: inline-block;
    width: 150px;
    height: 75px;
    margin: 10px;
    border: 3px solid #73AD21;
}
```

CSS Liquid Layout

```
body {
  background-color:#ffffff;}

div#page {
  width:80%;
  padding:10%;
  background-color:#ffffff;
  border:1px solid #000000;
  font-family:arial, verdana, sans-serif;
  font-size:12px;
}
```

```
<body>
   <div id="page">

<h1>Sample Web Page</h1>
<p>Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. </p>

</div>
```

CSS max-width

```
body {
  background-color:#efefef;}

div#page {
  max-width:880px;
  padding:20px;
  margin:20px;
  margin-left:auto;
  margin-right:auto;
  background-color:#ffffff;
  border:1px solid #000000;
  font-family:arial, verdana, sans-serif;
  font-size:12px;
}
```

```
<body>
   <div id="page">

<h1>Sample Web Page</h1>
<p>Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. </p>

</div>
```

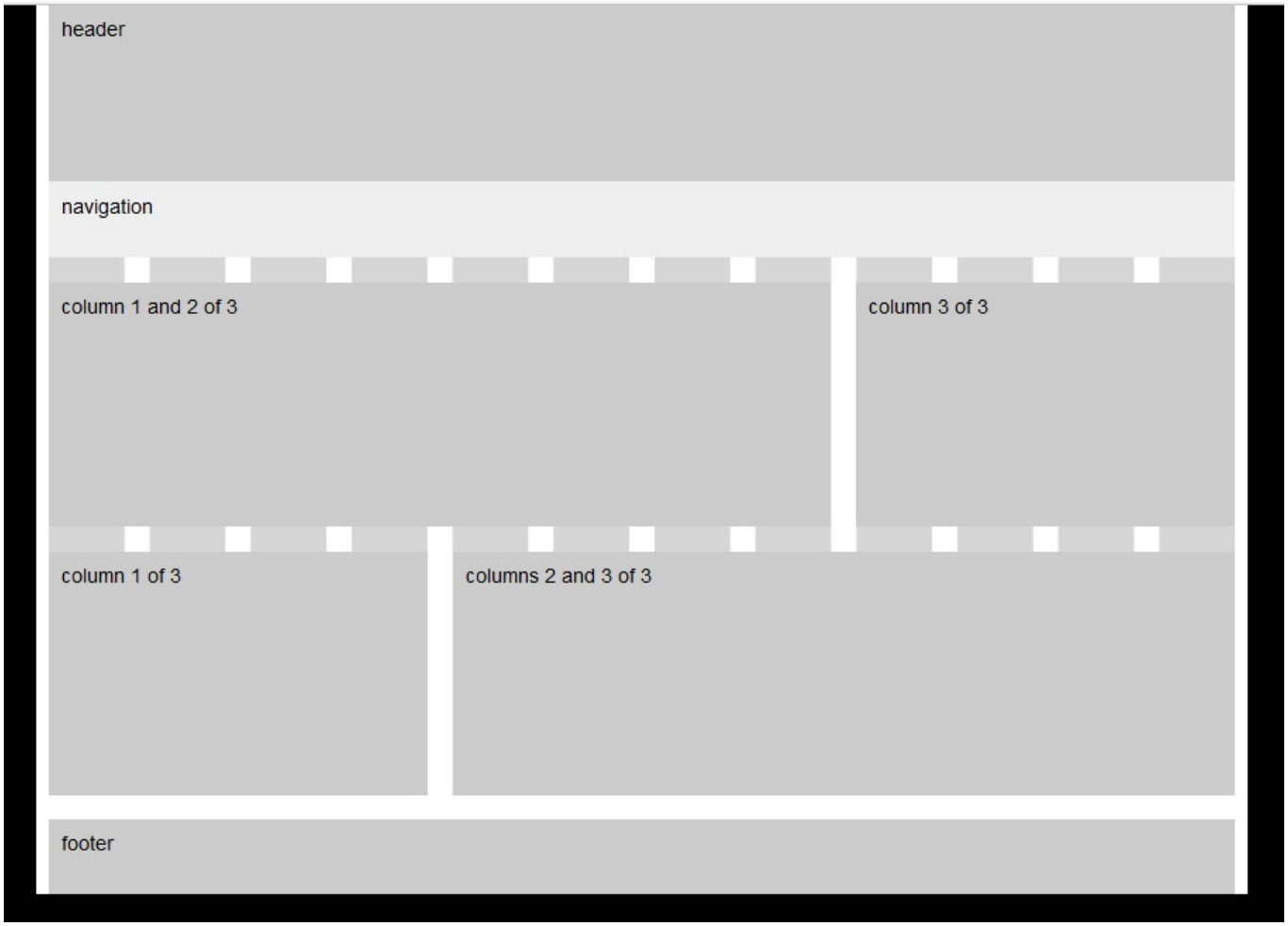
CSS image grid

```
body {
    background-color:#383838;
    margin:0px;
    padding:0px;}
h1 {
    padding : 5%;
    margin:0px;}


#frame {
  width:960px;
  margin-left:auto;
  margin-right:auto;
  background-image:url("../images/960px_12_col_grid.gif");
  background-repeat:repeat-y;}

#page {
  width:940px;
  margin-left:auto;
  margin-right:auto;}
```

```
<body>
<div id="frame">
      <div id="page">
      <h1>Testing the background grid</h1>









      </div>
</div>
</body>
```

Sample column layout

Sample column layout (2)



Sample column layout (3)