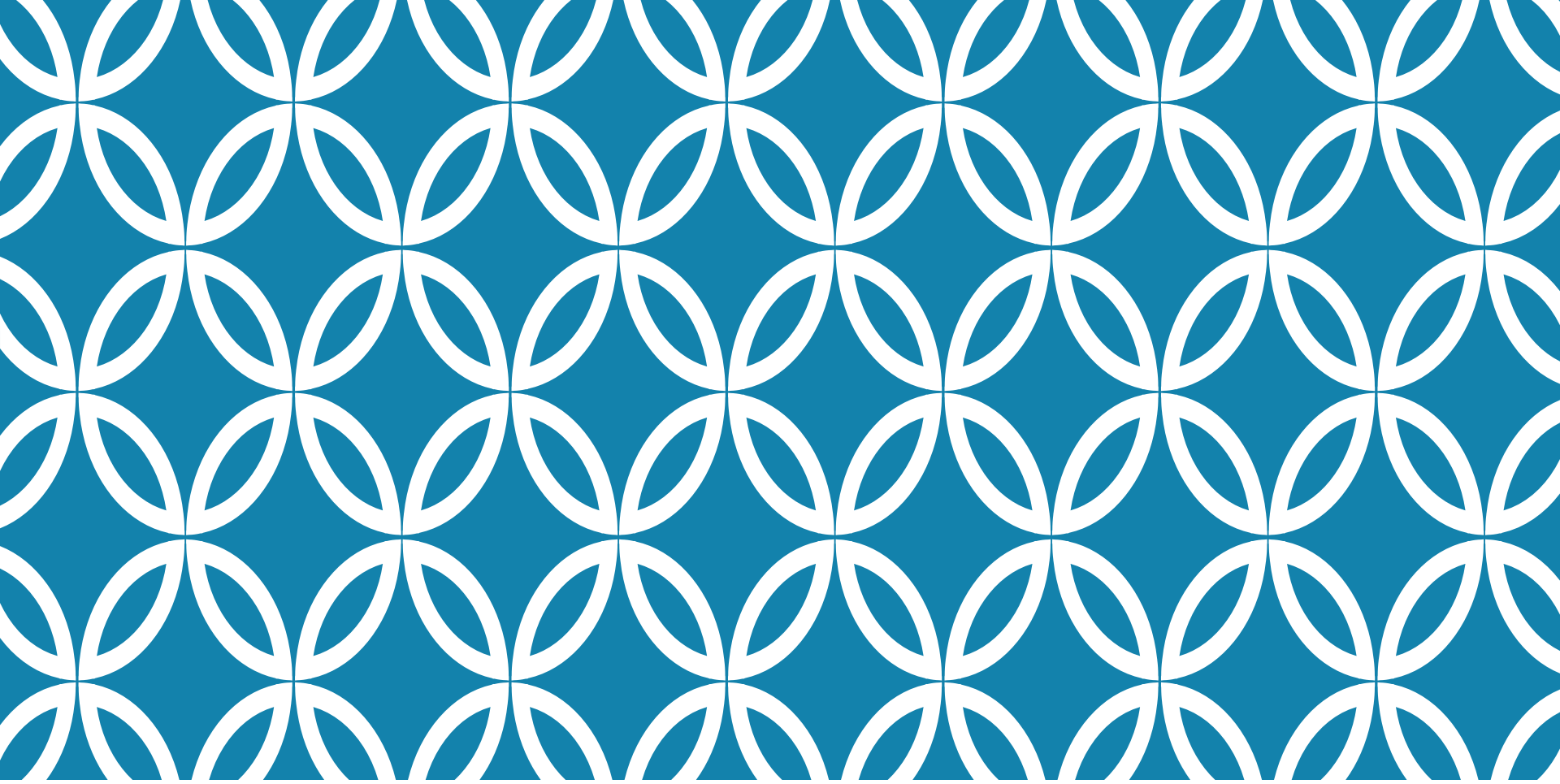


CSS LAYOUTS

BLOCK LEVEL ELEMENTS

Block-level elements such as <p>, , <h2>, <ul>, and <table> are each contained on their own line. Because block-level elements begin with a line break.

Without styling, two block-level elements can't exist on the same line

BLOCK LEVEL ELEMENTS

Each block exists on its own line and is displayed in normal flow from the browser window's top to its bottom.

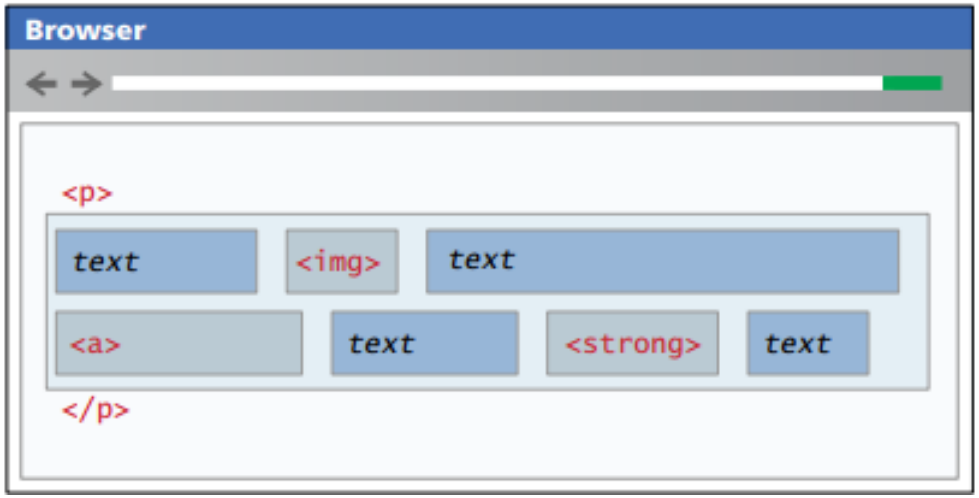
By default each block-level element fills up the entire width of its parent (in this case, it is the <body>, which is equivalent to the width of the browser window).

You can use CSS box model properties to customize, for instance, the width of the box and the margin space between other block-level elements.

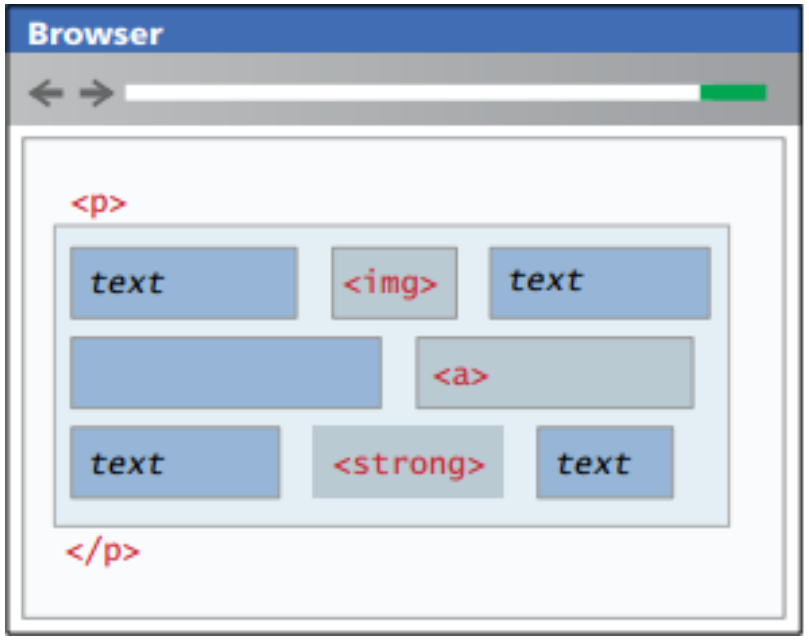
INLINE ELEMENTS

Inline elements do not form their own blocks but instead are displayed within lines. Normal text in an HTML document is inline, as are elements such as , <a>, <img>, and . Inline elements line up next to one another horizontally from left to right on the same line

```
<p>
This photo <img src="photo-con.png" alt="" /> of Conservatory Pond in
<a href="http://www.centralpark.com/">Central Park</a> New York City
was taken on October 22, 2015 with a <strong>Canon EOS 30D</strong>
camera.
</p>
```



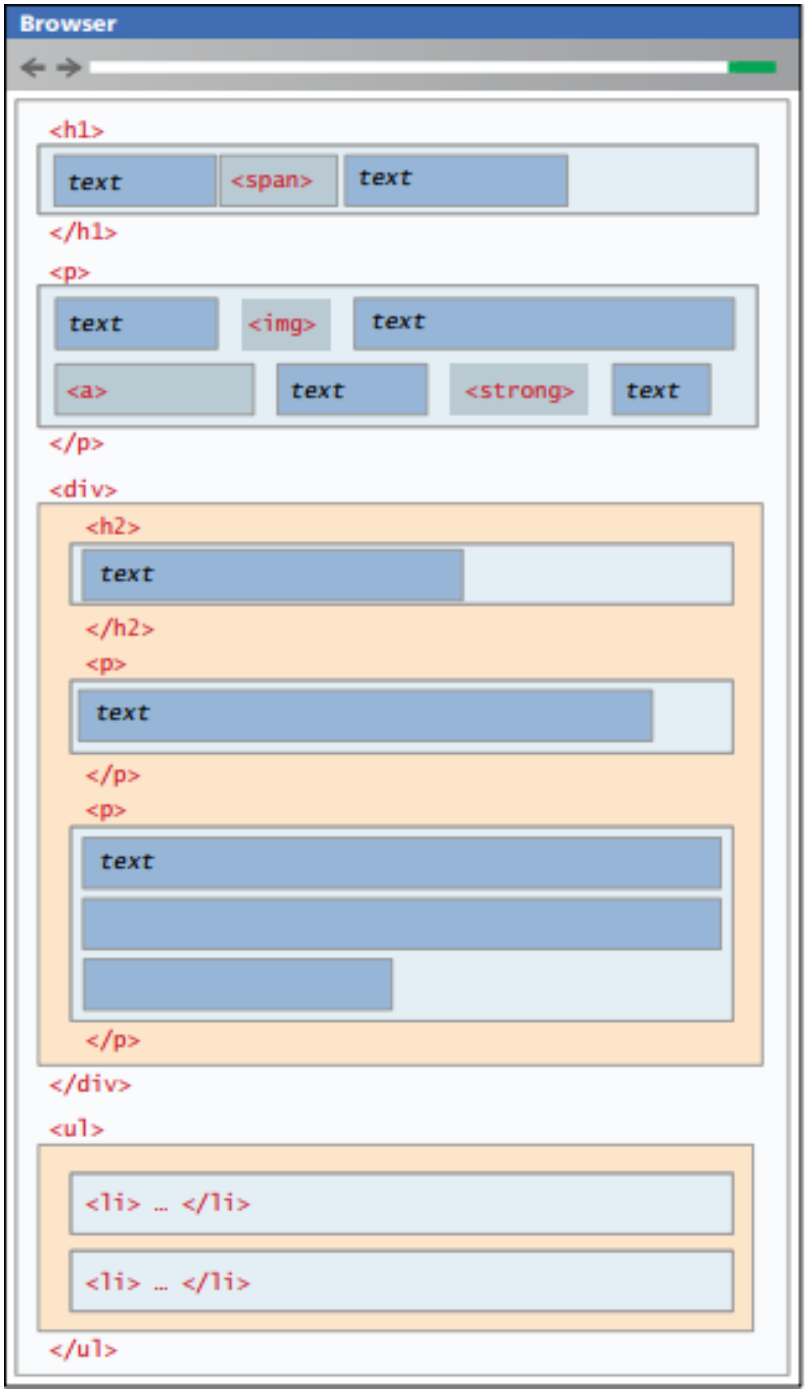
INLINE ELEMENTS – REFLOWED



If the browser window resizes, then inline content will be "reflowed" based on the new width.

Here the content of this <p> element is now displayed on three lines.

BLOCK AND INLINE ELEMENTS



A document consists of block-level elements stacked from top to bottom.

Within a block, inline content is horizontally placed left to right.

Some block-level elements can contain other block-level elements (in this example, a `<div>` can contain other blocks).

In such a case, the block-level content inside the parent is stacked from top to bottom within the container (`<div>`).

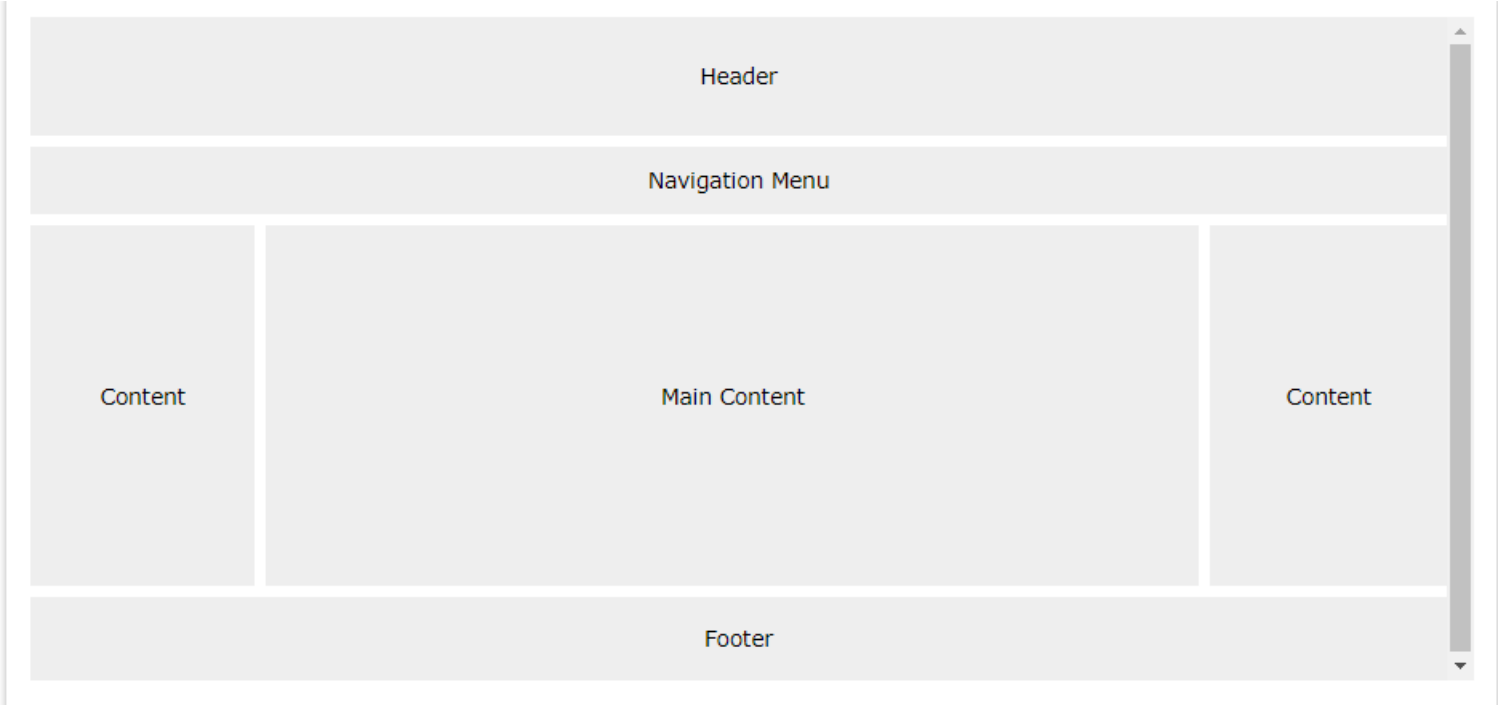
BLOCK TO INLINE & INLINE TO BLOCK

It is possible to change whether an element is block-level or inline via the CSS display property. Consider the following two CSS rules:

```
span { display: block; }
li { display: inline; }
```

These two rules will make all elements behave like block-level elements and all <li> elements like inline (that is, each list item will be displayed on the same line)

USUAL WEBSITE LAYOUT



CSS LAYOUTS

CSS page layout techniques allow us to take elements contained in a web page and control where they are positioned

- relative to their default position in normal layout flow,
- the other elements around them,
- their parent container,
- or the main viewport / window.

TYPES OF LAYOUTS

Normal flow

The display property

Flexbox

Grid

Floats

Positioning

Table layout

Multiple-column layout

1. NORMAL FLOW

Normal flow is how the browser lays out HTML pages by default when you do nothing to control page layout

The elements that appear one below the other are described as *block* elements, in contrast to *inline* elements, which appear one beside the other, like the individual words in a paragraph.

```
<p>I love my cat.</p>

<ul>
  <li>Buy cat food</li>
  <li>Exercise</li>
  <li>Cheer up friend</li>
</ul>

<p>The end!</p>
```

By default, the browser will display this code as follows:

I love my cat.

- Buy cat food
- Exercise
- Cheer up friend

The end!

METHODS TO CHANGE LAYOUT

The display property – Standard values such as block, inline or inline- block can change how elements behave in normal flow

Floats – Applying a float value such as left can cause block level elements to wrap alongside one side of an element, like the way images sometimes have text floating around them in magazine layouts.

The position property – Allows you to precisely control the placement of boxes inside other boxes. static positioning is the default in normal flow, but you can cause elements to be laid out differently using other values, for example always fixed to the top left of the browser viewport.

Table layout – features designed for styling the parts of an HTML table can be used on non-table elements using display: table and associated properties.

Multi-column layout – The Multi-column layout properties can cause the content of a block to layout in columns, as you might see in a newspaper.

2. DISPLAY

Display:

As seen before

Block

Block takes up the **whole width by default**

Width = 60%

Width = 20%

Even if there is enough space, block elements do not "sit" next to each other (unless if they are floated)

Inline

Inline elements are things like **bold** and *italics*.
You cannot control the height and width of these elements

Inline-block

inline block elements can sit on the same line

width = 33%	width = 33%	width = 33%

AND

combines | block properties | and *inline* properties

3. FLEXBOX

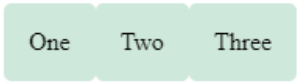
Flexbox is the short name for the Flexible Box Layout Module, designed to make it easy for us to lay things out in one dimension – either as a row or as a column.

The initial value of flex-direction is row.

```
.wrapper {
  display: flex;
  flex-direction: column;
}
```

```
.wrapper {
  display: flex;
}
```

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
</div>
```

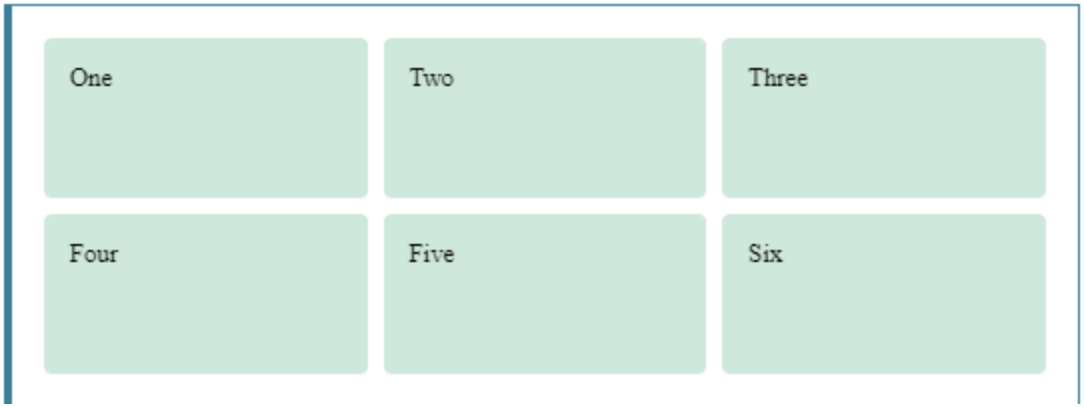


4. GRID LAYOUT

While flexbox is designed for one-dimensional layout, Grid Layout is designed for two dimensions – lining things up in rows and columns.

```css
.wrapper {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: 100px 100px;
    grid-gap: 10px;
}
```

```html
<div class="wrapper">
    <div class="box1">One</div>
    <div class="box2">Two</div>
    <div class="box3">Three</div>
    <div class="box4">Four</div>
    <div class="box5">Five</div>
    <div class="box6">Six</div>
</div>
```



UNEQUAL GRID

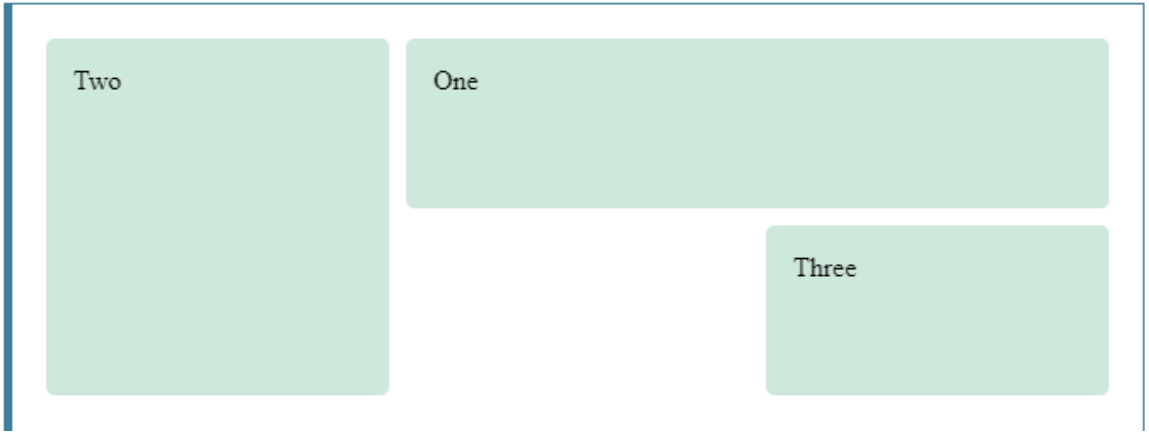
```css
.wrapper {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: 100px 100px;
    grid-gap: 10px;
}

.box1 {
    grid-column: 2 / 4;
    grid-row: 1;
}

.box2 {
    grid-column: 1;
    grid-row: 1 / 3;
}

.box3 {
    grid-row: 2;
    grid-column: 3;
}
```

```html
<div class="wrapper">
    <div class="box1">One</div>
    <div class="box2">Two</div>
    <div class="box3">Three</div>
</div>
```



5. FLOATS

Floating an element changes the behavior of that element and the block level elements that follow it in normal flow.

- The element is moved to the left or right and removed from normal flow, and the surrounding content floats around the floated item.

The float property has four possible values:

left – Floats the element to the left.

right – Floats the element to the right.

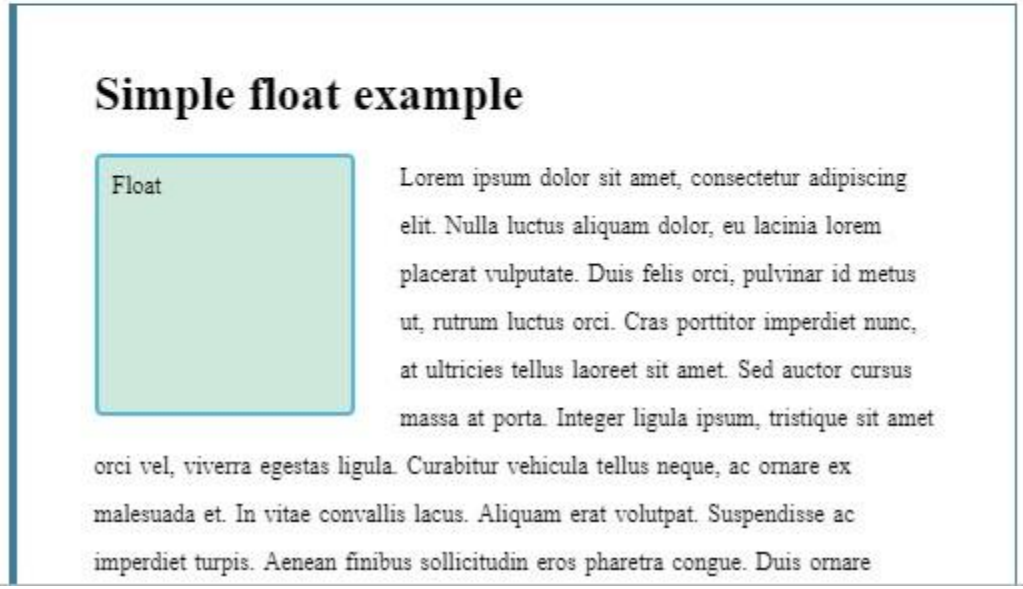
inherit – Specifies that the value of the float property should be inherited from the element's parent element.

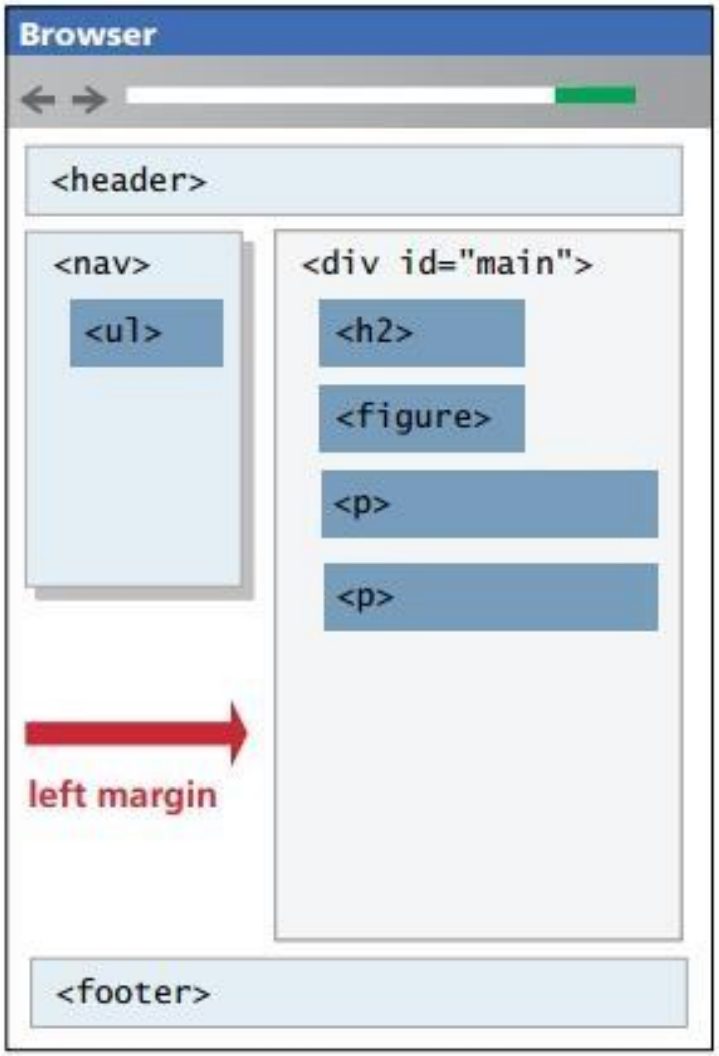
```html
<h1>Simple float example</h1>

<div class="box">Float</div>

<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu la
```

```css
.box {
    float: left;
    width: 150px;
    height: 150px;
    margin-right: 30px;
}
```






```
div#main {
  …
  margin-left: 220px;
}
```

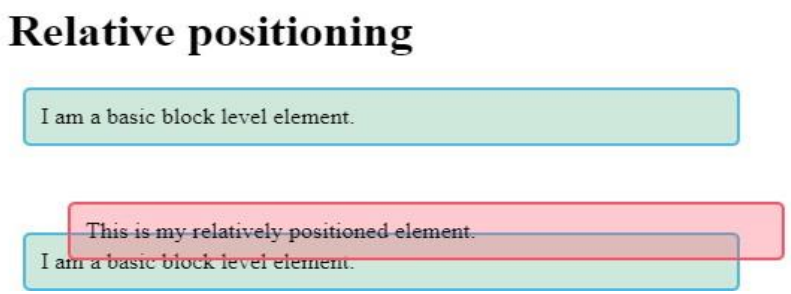
6. POSITIONING TECHNIQUES

Relative positioning

allows you to modify an element's position on the page, moving it relative to its position in normal flow

– including making it overlap other elements on the page.

```
.positioned {
  position: relative;
  background: rgba(255,84,104,.3);
  border: 2px solid rgb(255,84,104);
  top: 30px;
  left: 30px;
}
```

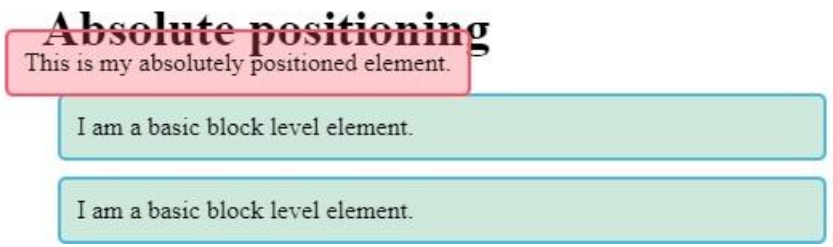


6. POSITIONING TECHNIQUES

Absolute positioning moves an element completely out of the page's normal layout flow

- a position relative to the edges of the page's <html> element (or its nearest positioned ancestor element).

```
.positioned {
    position: absolute;
    background: rgba(255,84,104,.3);
    border: 2px solid rgb(255,84,104);
    top: 30px;
    left: 30px;
}
```



6. POSITIONING TECHNIQUES

Fixed positioning is very similar to absolute positioning, except that

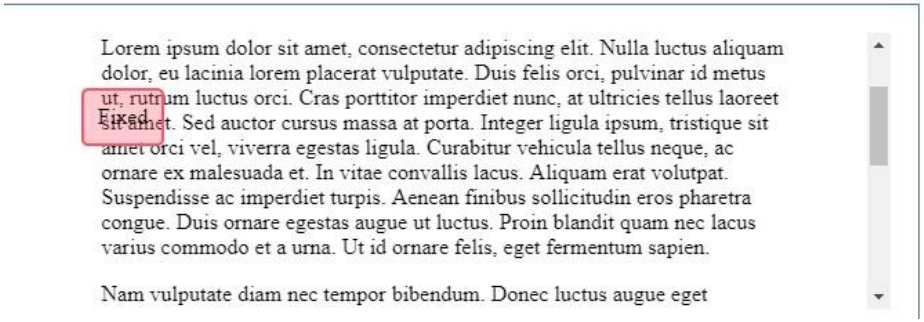
- it fixes an element relative to the browser viewport, not another element.
- This is useful for creating effects such as a persistent navigation menu that always stays in the same place on the screen as the rest of the content scrolls.

```
<h1>Fixed positioning</h1>

<div class="positioned">Fixed</div>

<p>Paragraph 1.</p>
<p>Paragraph 2.</p>
<p>Paragraph 3.</p>
```

```
.positioned {
    position: fixed;
    top: 30px;
    left: 30px;
}
```

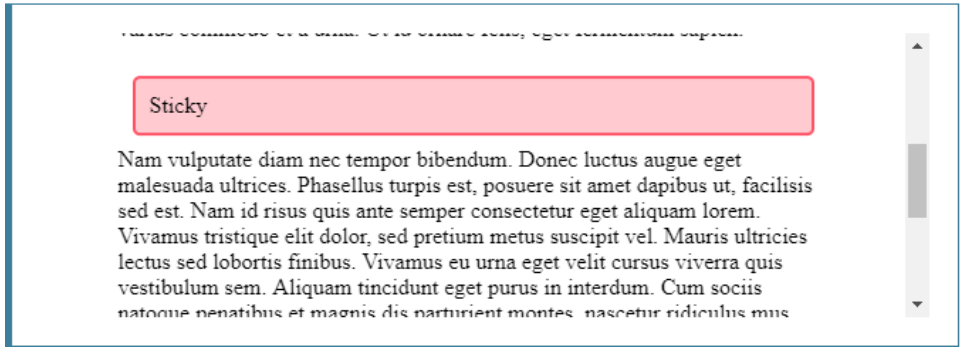


6. POSITIONING TECHNIQUES

Sticky positioning

When an item has position: sticky it will scroll in normal flow until it hits offsets from the viewport that we have defined. At that point it becomes "stuck" as if it had position: fixed applied.

```
1  .positioned {
2    position: sticky;
3    top: 30px;
4    left: 30px;
5  }
```



7. TABLE LAYOUT

The way that a table looks on a webpage when you use table markup is due to a set of CSS properties that define table layout.

These properties can be used to lay out elements that are not tables, a use which is sometimes described as "using CSS tables".

```html
<form>
  <p>First of all, tell us your name and age.</p>
  <div>
    <label for="fname">First name:</label>
    <input type="text" id="fname">
  </div>
  <div>
    <label for="lname">Last name:</label>
    <input type="text" id="lname">
  </div>
  <div>
    <label for="age">Age:</label>
    <input type="text" id="age">
  </div>
</form>
```

This gives us the following result:


```css
html {
  font-family: sans-serif;
}

form {
  display: table;
  margin: 0 auto;
}

form div {
  display: table-row;
}

form label, form input {
  display: table-cell;
  margin-bottom: 10px;
}

form label {
  width: 200px;
  padding-right: 5%;
  text-align: right;
}

form input {
  width: 300px;
}

form p {
  display: table-caption;
  caption-side: bottom;
  width: 300px;
  color: #999;
  font-style: italic;
}
```

8. MULTI-COLUMN LAYOUT

The multi-column layout module gives us a way to lay out content in columns, similar to how text flows in a newspaper.

We are using a column-width of 200 pixels on that container, causing the browser to create as many 200-pixel columns as will fit in the container and then share the remaining space between the created columns.

Column-width

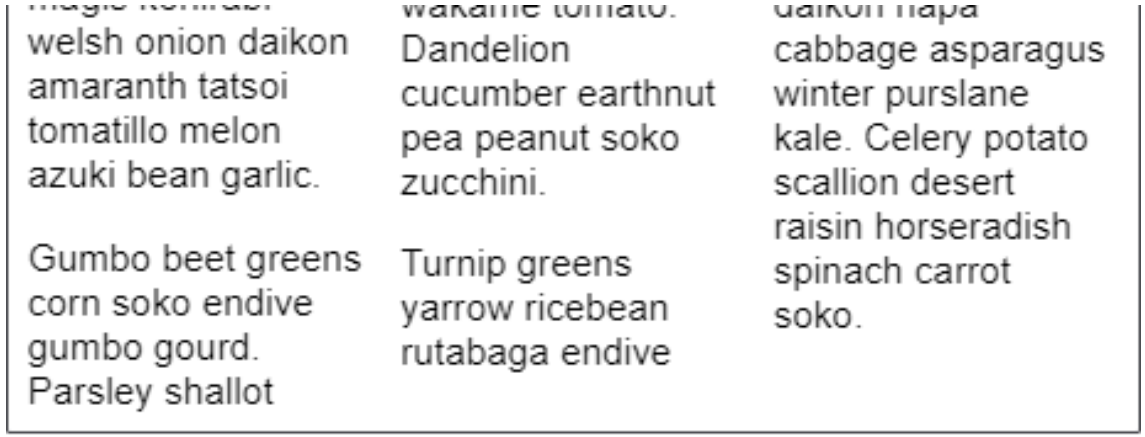
```html
<div class="container">
    <h1>Multi-column layout</h1>

    <p>Paragraph 1.</p>
    <p>Paragraph 2.</p>

</div>
```

```css
.container {
        column-width: 200px;
    }
```

Column-count:




```
.container {
  column-count: 3;
}
```

```
  <p>Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot
courgette tatsoi pea sprouts fava bean collard greens dandelion okra
wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.
</p>

  <p>Turnip greens yarrow ricebean rutabaga endive cauliflower sea
```

Reference: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Introduction