HTML devides the code into sections.

These sections are defined by so-called tags.

Some tags define the role text plays on the page. For example the **<h1>** tag is the most important introduction to a page's content. Other headers let you divide the content into other, less important, but related sections such as the **<h2>**, **<h3>** tag or a tag identifying a paragraph ** **.

Each tag has an opening and a closing tag, for example ** ** for the opening and ** ** for the closing tag. In between the tags is the actual, visible content.

Tags can be nested as in the list example on the right. The list is defined be the tag **<ul>** ("unordered list") and the list item tags **(<li>**) are nested inside.

There are generic tags such as the ** ** tag which has no effect on the appearance of text. It's used to style certain parts or divisions in the layout such as header, main content and footer areas.

A CSS file is a separte document that contains style definitions for those HTML tags.

HTML source code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://www.
w3.org/1999/xhtml">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Caprese Salad</title>
</head>

<body>

<h1>Caprese Salad</h1>

<p>Ready in: 15 Min</p>

<h2>Ingredients</h2>

<ul>
<li>2 large ripe tomatoes, peeled and sliced 1/4-inch thick</li>
<li>8 ounces fresh mozzarella, sliced 1/4-inch thick</li>
<li>1/4 teaspoon salt</li>
<li>1/4 teaspoon freshly ground black pepper</li>
<li>2 tablespoons extra-virgin olive oil</li>
<li>8 fresh basil leaves</li>
</ul>

<h2>Directions</h2>

<p>Arrange the tomato and mozzarella slices on a platter or individual salad plates,
overlapping the slices and fanning them out like a deck of cards.</p>

<p>Sprinkle with the salt and pepper.</p>

<p>Drizzle with the oil.</p>

<p>Garnish with the basil: Cut it into very thin slices or tear into bits and sprinkle on
top or leave the leaves whole and tuck them here and there between the mozzarella
and tomato slices. Serve immediately.</p>

<h3>Restaurant Recipe</h3>

<p>This recipe was provided by professional chefs and has been scaled down from
a bulk recipe provided by a restaurant. The Food Network Kitchens chefs have not
tested this recipe, in the proportions indicated, and therefore, we cannot make any
representation as to the results.</p>

</body>

</html>
```

HTML page in browser (without CSS styles)

Caprese Salad

Ready in: 15 Min

Ingredients

- 2 large ripe tomatoes, peeled and sliced 1/4-inch thick
- 8 ounces fresh mozzarella, sliced 1/4-inch thick
- 1/4 teaspoon salt
- 1/4 teaspoon freshly ground black pepper
- 2 tablespoons extra-virgin olive oil
- 8 fresh basil leaves

Directions

Arrange the tomato and mozzarella slices on a platter or individual salad plates, overlapping the slices and fanning them out like a deck of cards.

Sprinkle with the salt and pepper.

Drizzle with the oil.

Garnish with the basil: Cut it into very thin slices or tear into bits and sprinkle on top or leave the leaves whole and tuck them here and there between the mozzarella and tomato slices. Serve immediately.

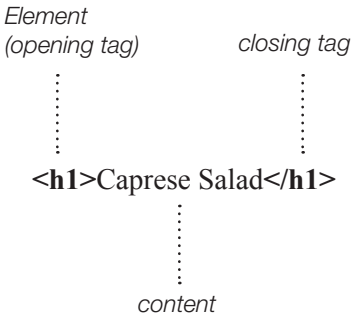
Restaurant Recipe

This recipe was provided by professional chefs and has been scaled down from a bulk recipe provided by a restaurant. The Food Network Kitchens chefs have not tested this recipe, in the proportions indicated, and therefore, we cannot make any representation as to the results.

HTML in browser (without CSS)

Caprese Salad

HTML Code

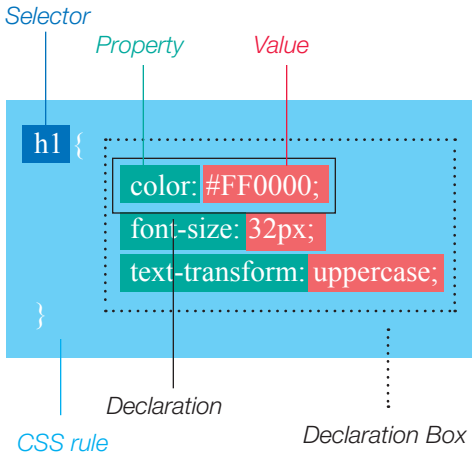


Element: The “atomic pieces” of the document. For example, h1, h2, p, ul

They always start with an opening and closing tag.

These Elements can be styled with CSS.

CSS



Selector: Defines the elements of a document that have declarations applied to them

Property: Describes an aspect of an element/tag’s presentation such as color, font-size or borders.

Value: Is a descriptor defining a specific appearance, such as coilor name or color.

Declaration: Is a property-value pair.

Declaraton Block: Is a set of declarations grouped together

Rule: Is a pairing of a selector and a declaration block.

Style Sheet: Is a collection of rules that will be applied to an HTML document

HTML in browser (with CSS styles)

CAPRESE SALAD

HTML in browser (without CSS)

Ingredients

- 2 large ripe tomatoes, peeled and sliced
- 8 ounces fresh mozzarella, sliced
- 1/4 teaspoon salt
- 1/4 teaspoon freshly ground black pepper
- 2 tablespoons extra-virgin olive oil
- 8 fresh basil leaves

HTML Code

```
<body>

<div id=ingredients>
<h2>Ingredients</h2>
<ul >
   <li>2 large ripe <span class=red>tomatoes</span>,
   peeled and sliced</li>
   <li>8 ounces fresh mozzarella, sliced</li>
   <li>1/4 teaspoon salt</li>
   <li>1/4 teaspoon freshly ground black pepper</li>
   <li>2 tablespoons extra-virgin olive oil</li>
   <li>8 fresh basil leaves</li>
   <li></li>
</ul>
</div>

</body>
```

Tag Selectors: *Page -wide Styling.* Sometimes also called Element Selectors. For example: body, h1, h2, ul, li, p

The "**div**" tag defines a division of the page. If there's no other tag selector in the HTML to style use "div" or "span" tags.

The "**span**" tag is useful for styling "pieces" nested insite other tags (one word in a paragraph, for example).

Class Selectors: *Pinpoint Control.* Used to style one or more elements that are different from related tags. Unlike the id selector, the class selector is most often used on several elements

ID Selectors: *Specific Page Sections.* Used to identifiy a unique part of a page.

CSS

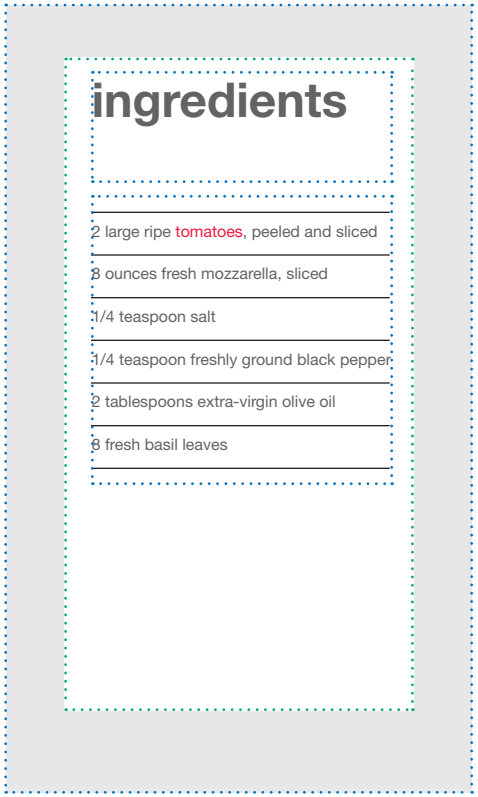
```
body {
        background: #CCC;
}
#ingredients {
        width: 230px;
        heigh: 500px;
        color: #999;
        background: #FFF;
        margin: 5px 10px 10px 10px;
        padding: 50px 0px 20px 10px;
}
h2 {
        font-size: 25px;
        text-transform: lowercase;
        padding-bottom: 25px;
}
ul {
        width: 200px;
        list-style: none;
        padding: 0;
}
li {
        border-top: 1px solid #000;
        border-bottom: none;
        margin: 5px 0px 5px 0px;
        padding: 5px 0px 0px 0px;
}
.red {
        color: red;
}
```

Tag Selectors: predefined names such as body, h1, h2 match corresponding tag inside HTML code

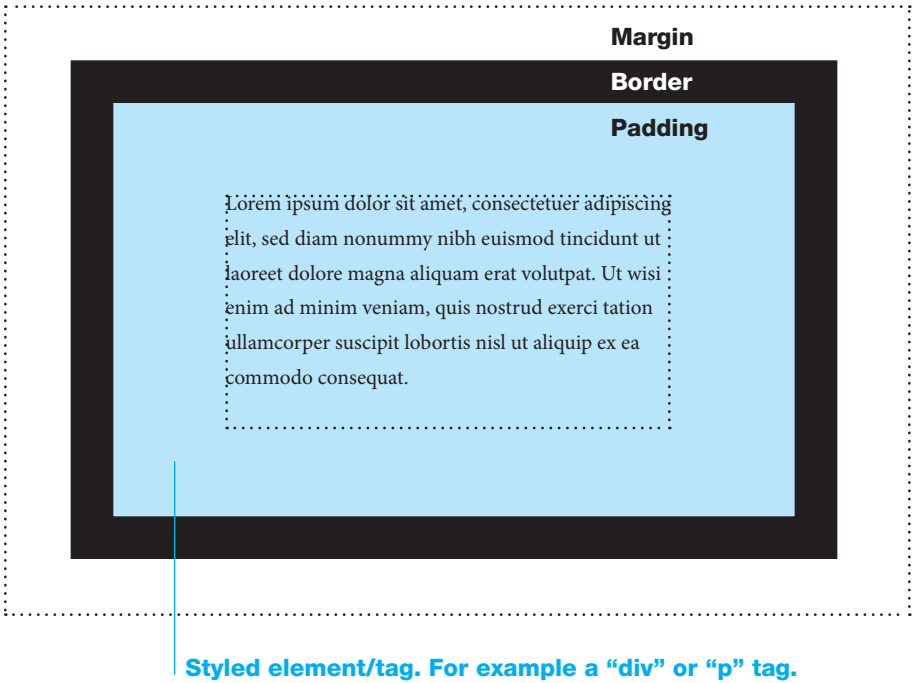
Class Selectors: start with a period inside the CSS style sheet. You need to give them a name.

ID Selectors: start with a pound symbol inside the CSS style sheet. You can give them a name of your choice.

HTML in browser (with CSS styles)



CSS Box Model



All HTML elements can be considered as boxes. In CSS, the term “box model” is used when talking about design and layout.

The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.

The box model allows us to place a border around elements and space elements in relation to other elements.

If left and right margin of an element is set to “auto” content will be centered.

Margin - Clears an area around the border. The margin does not have a background color, and it is completely transparent

Border - A border that lies around the padding and content. The border is affected by the background color of the box

Padding - Clears an area around the content. The padding is affected by the background color of the box

Content - The content of the box, where text and images appear

Width and Height of an Element

Important: When you specify the width and height properties of an element with CSS, you are just setting the width and height of the content area. To know the full size of the element, you must also add the padding, border and margin.

The total **width** of an element should always be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should always be calculated like this:

Total element **height** = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Example:

The total width of the paragraph element in the example below is 300px:

```
p {
      width:250px;
      padding:10px;
      border:5px solid gray;
      margin:10px;
}
```

Let’s do the math:
250px (width)
+ 20px (left and right padding)
+ 10px (left and right border)
+ 20px (left and right margin)
= 300px

Padding Properties:

padding-top
padding-bottom
padding-right
padding-left

To shorten the code, it’s possible to combine properties. This is called a “shorthand property”. For example:

```
padding-top:25px;
padding-right:50px;
padding-bottom:25px;
padding-left:50px;
```

Shorthand property (top, right, bottom, left):

```
padding: 25px 50px 25px 50px;
```

Margin Properties:

margin-top
margin-bottom
margin-right
margin-left

Border Properties:

border-top
border-right
border-bottom
border-left

border-color

border-style
(values are for example: none, hidden, dotted, dashed, solid)

CSS Font Properties

Property	Description	Values
font	Sets all the font properties in one declaration	font-family font-size font-weight font-style font-variant
font-family	Specifies the font family for text	family-name generic-family
font-size	Specifies the font size of text	px % xx-small x-small small medium large x-large xx-large smaller larger length
font-style	Specifies the font style for text	normal italic oblique
font-variant	Specifies whether or not a text should be displayed in a small-caps font	normal small-caps
font-weight	Specifies the weight of a font	normal bold bolder lighter 100 -900

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts
Sans-serif	Arial Verdana	Sans-serif fonts
Monospace	Courier New Lucida Console	All monospace characters have the same wtdth

CSS Text Properties

Property	Description	Values
color	Sets the color of a text	color
direction	Sets the text direction	ltr rtl
line-height	Sets the distance between lines	normal number length %
letter-spacing	Increase or decrease the space between characters	normal length
text-align	Aligns the text in an element	left right center justify
text-decoration	Adds decoration to text	none underline overline line-through blink
text-indent	Indents the first line of text in an element	length %
text-transform	Controls the letters in an element	none capitalize uppercase lowercase
vertical-align	Sets the vertical alignment of an element	baseline sub super top text-top middle bottom text-bottom length %
word-spacing	Increase or decrease the space between words	normal length

Simple CSS Properties (for a complete list with explanations go to www.w3schools.com/css)

positioning Properties:

padding
padding-top
padding-right
padding-bottom
padding-left

shorthand:
padding: top right bottom left;

example:
padding: 10px 20px 10px 20ox;

margin
margin-top
margin-right
margin-bottom
margin-left

same shorthand as above. Just replace "padding" with "margin".

also experiment with negative values.
example:
margin-left: -200px;

float
float: left
flloat: right

clear (to clear floats)
clear: left
clear: right
cleat: both

border properties:

border-style
(this property has to be set to make the border visible. Values are for example: none, hidden, dotted, dashed, solid)

border-top
border-right
border-bottom
border-left

border-color

border-width

border-top-style
border-right-style
border-bottom-style
border-left-style

shorthand example:
border: 5px solid #C30;

or if you only want a top border to appear:
border-top: 5px solid #C30;

When using the border property, the order of the values are:

* border-width
* border-style
* border-color

It does not matter if one of the values above are missing (although, border-style is required), as long as the rest are in the specified order.

color and background color properties:

color
The "color" property will give the content of the element a color such as text, for example

example:
color: red:
or
color: #FF0000;
or
color: rgb(255,255,255):

background-color
this property will basically color the element itself

example:
background-color: red:
or
background -color: #FF0000;

text properties

font-family
fonts have to be installed on the computer of the person visiting a website. So you are restricted to only a few fonts that most likely are installed on a computer. For now stick with the options Dreamweaver gives you when typing in the font-family property in your CSS file.

examples:
font-family: Arial, Helvetica, sans-serif;

font-family: "Times New Roman", Times, serif;

font-size
For the values there are plenty of sizing units to choose from: keywords, ems, exs, pixels, percentages, picas, points, inches, centimeters, millimeters. Let's only use the pixel unit for the first assignment.

example:
font-size: 24px;

text-transform
(values are uppercase, lowercase)
text-decoration
(values are underline, overline, line-through)
line-height
(specify values in pixels for now)
letter-spacing
(specify values in pixels for now)
letter-spacing
(specify values in pixels for now)
word-spacing
(specify values in pixels for now)

Background Property

CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

background-color
background-image
background-repeat
background-attachment
background-position

For the first assignment we won't use images yet and focus only on the first propety, the "background-color".

The background color can be specified by:

* name - a color name, like "red"
* RGB - an RGB value, like "rgb(255,0,0)"
* Hex - a hex value, like "#ff0000"

The background color of a page is defined in the body selector:

```
body {
        background-color:#999999;
}
```

Difference between the "color" and the "background-color" property:

In the example below the text of the paragraph is blue and the background color is yellow. So the "color" property defines the contents of the element the "background-color" property it's background.

Lorem ipsum dolor sit amet

The CSS code would be:

```
p {
        color: blue;
        background-color: yellow;
}
```

By the way:
It doesn't really matter how you write your CSS rules. The example on top could also be written the following way:

```
p {color: blue; background-color: yellow;}
```

The space after the "p" is important as well as the colons and semi-colons after the properties and the values. How you organize the rule is up to your preference. If you have many rules in one style sheet file the example on top is probably the better one since the individual declarations are easier to visually separate from each other.

CSS Floats

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.

Float is very often used for images, but it is also useful when working with layouts.

How Elements Float

Elements are floated horizontally, this means that an element can only be floated left or right, not up or down.

A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.

The elements after the floating element will flow around it.

The elements before the floating element will not be affected.

Float values: *left, right, both, none, inherit*

Floating Elements Next to Each Other

If you place several floating elements after each other, they will float next to each other if there is room.

Turning off Float - Using Clear

Elements after the floating element will flow around it. To avoid this, use the clear property.

The clear property specifies which sides of an element other floating elements are not allowed.

Clear values: *left, right, both, none, inherit*

Here an example of how to define a rule for a layout with 3 columns (CSS = blue, HTML = green):

```
.column {
        float: left;
        width: 200px;
        margin: 20px;
}
```

```
<body>
<div class="column"><p>Lorem ipsum dolor.</p>
</div>
<div class="column"><p>consectetuer diam</p>
</div>
<div class="column"><p>sed diam nonummy</p>
</div>'
</body>
```

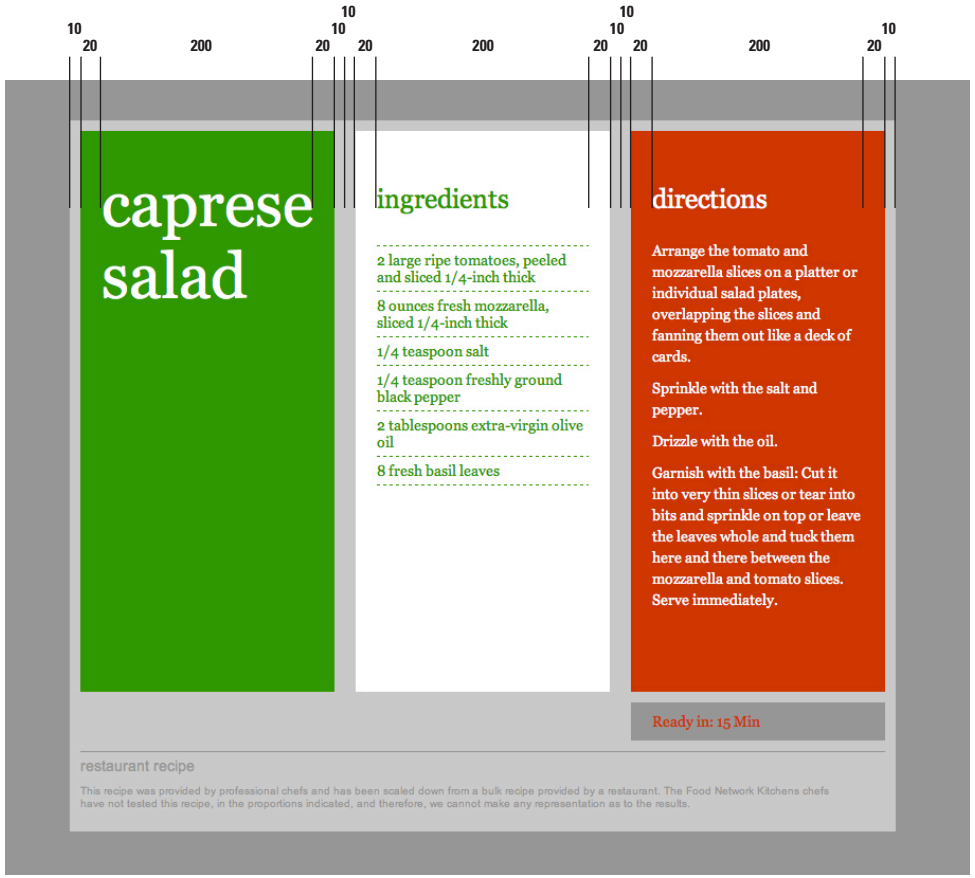
Multi Column Layout using floated elements

In the example below floats are applied to 4 elements defined by div tags:
The title, ingredients, directions and the cooking time.

When floating elements side by side you have to pay attention to the measurements since margin and paddings are added to the widths of the elements. Each floated column is set to 200px width with a left and right margin of 10px and a left and right padding of 20px.

In order to avoid having the red column drop below the first two make sure the total width doesn't exceed our limit of 780px (add margin, padding and border width to the total!).

Here is the math: 10+20+200+20+10+10+20+200+20+10+10+20+200+20+10 **= 780**



HTML order of elements:

When you work with floats, the order of the HTML code is important. The HTML for the floated element must appear *before* the element that wraps around it.

In our simple example the HTML order is:

Title, ingredients and directions are floated to the left, the cooking time div is floated to the right.

A footnote is usually something that should appear all the way at the bottom of a page. In order to avoid this element to wrap around any other element the *clear* property can be applied to this element (see code on the right: clear: both;).

HTML (simplified)

```
<body>
<div id=wrapper>
<div id=title>
</div>
<div id=title>
</div>
<div id=directions>
</div>
<div id=time>
</div>
<div id=footnote>
</div>
</div>
</body>
```

CSS (simplified)

```
#title {
        float: left;
        width: 200px;
        margin: 10px 10px 10px 10px;
        padding: 50px 20px 0px 20px; }
#ingredients {
        float: left;
        width: 200px;
        margin: 10px 10px 10px 10px;
        padding: 50px 20px 0px 20px; }
#directions {
        float: left;
        width: 200px;
        margin: 10px 10px 10px 10px;
        padding: 50px 20px 0px 20px; }
#time {
        float: right;
        width: 200px;
        margin: 10px 10px 10px 10px;
        padding: 10px 20px 10px 20px; }
#footnote {
        clear: both;
        margin: 10px 10px 10px 0px;
        padding: 10px 20px 0px 0px;
        border-top: 1px solid #999; }
```

Positoning Block Elements

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

<h1>, <p>,

Center Aligning Using the margin Property

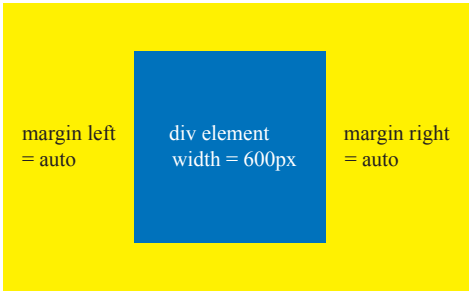
Block elements can be aligned by setting the left and right margins to “auto”.

Note: Using margin:auto will not work in Internet Explorer, unless a !DOCTYPE is declared.

Setting the left and right margins to auto specifies that they should split the available margin equally. The result is a centered element (CSS = blue, HTML = green):

```
.center {
        margin-left:auto;
        margin-right:auto;
        width:600px;
}

<body>
<div class=”center”>
</div>
</body>
```



Left and Right Aligning Using the position Property

One method of aligning elements is to use absolute positioning:

Example

```
.right {
        position:absolute;
        right:0px;
        width:300px;
}

<body>
<div class=”right”>
</div>
</body>
```

The CSS position properties may be more difficult to understand and to control than positioning elements with margin and padding properties. General rule for now: Use position properties if you can’t achieve it with margins, paddings and floats.

Left and Right Aligning Using the float Property

One method of aligning elements is to use the float property:

Example

```
.right {
        float:right;
        width:300px;
        background-color:#b0e0e6;
}

<body>
<div class=”right”>
</div>
</body>
```

Positioning

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element’s content is too big.

Elements can be positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the positioning method.

There are four different positioning methods.

Static Positioning

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.

Static positioned elements are not affected by the top, bottom, left, and right properties.

Fixed Positioning

An element with fixed position is positioned relative to the browser window.

It will not move even if the window is scrolled:

Example

```
p.pos_fixed {
        position:fixed;
        top:30px;
        right:5px;
}
```

Relative Positioning

A relative positioned element is positioned relative to its normal position.

Example

```
h2.pos_left {
        position:relative;
        left:-20px;
}
```

Relatively positioned element are often used as container blocks for absolutely positioned elements.

Absolute Positioning

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

Example

```
h2 {
        position:absolute;
        left:100px;
        top:150px;
}
```

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.

Absolutely positioned elements can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order