

HyperText Markup Language

HTML&CSS



Unit 8

Web page
block layout.
Positioning.

Contents

Block Layout.....	3
Positioning of elements.....	16

Block Layout

layers

Despite the fact that the layer term is pretty often used at website development, detailed explanation of this term is rarely found in website development literature. Hereafter I will often use the layer term, so at first we need to define what it means.

Initially, the Netscape company introduced layers by adding the `<layer>` tag support to its browser. This tag allows hiding/showing current content, setting location relative to the browser window, putting layer over other layers, and loading data to the layer content from a file. All these parameters were changed using JavaScript, and it added options on creating really dynamic content on a page. Despite such impressive set of options, the `<layer>` tag was not included in the HTML specification and it remained a simple extension in the Netscape browser.

Although the necessity in the specified options became imminent and in the end of 1996 syntax for working with layers was developed and approved in working draft of the CSS Positioning (CSS-P) consortium. Styles took major load: they control appearance of any element, including changing values dynamically via JavaScript. Sadly, object models of browsers for access to elements differed, so a pretty complicated code that would take into account these special issues was to be written.

Now developers of popular browsers adhere to HTML and CSS specifications, which made site developers' lives easier since they need less time to test websites in various browsers.

Nevertheless, there are differences in approaches to browsers and in case they appear, developers adhere to the following.

If websites are slightly different in various browsers, a blind eye is turned to it. To be blunt, they are not fixed. Visitors won't pixel-wise compare a website in various browsers. Although mind that the website should be always displayed correctly and without errors.

If the website has significant differences when displaying in another browser, they are dealt using hacks.

Hack is a set of techniques when a certain browser receives a code that is perceived by this browser only, but ignored by the rest.

Despite that hacks work, it is better to use them to a limited extent or to do without them at all. The thing is, hacks decrease the universality of code, and in order to modify parameters of one element, a programmer needs to make changes in several places simultaneously.

There is another, perspective way: adhere to the CSS specification. Despite that browsers don't fully support it, they progress towards full support of various specifications (HTML, CSS, DOM). Thus, future browsers will be unified and they will display one and the same website correctly.

Let's get back to layers again. It is understood that they are directly related to styles. If so, is each element of the HTML code, to which styles are added, a layer? It is in a way. But we would be confused if we said layer instead of table or paragraph. So let's agree to apply this term to the `<div>` tags only.

In HTML4 and XHTML, layer is a web page element created using the `<div>` tag with a style design applied to it.

Thus, the block layout expression or layout using layers consists of constructive usage of the `<div>` tags and styles. At that, the following principles should be adhered.

Division of content and design

HTML code should contain only markup tags and logical formatting tags, and any design should be beyond the code, it should be in styles. Such approach allows managing appearance of elements and content of a page independently. Thanks to this, several people can work on a website, at that everyone works independently from each other. Designer, layout designer, and programmer work on their tasks on their own, which decreases website development time.

Using the `<div>` tag often

The `<div>` tag is very important in block layout, it accomplishes lots of functions. In fact, it's a base for styles that turn it to one thing or another. It does not mean that only this tag is applied, for we need to add images and design text. But when laying out by the means of layers, the `<div>` tag is a brick of layout, its foundation.

Thanks to this tag, the HTML code falls into several clear blocks, the code becoming more compact than at table layout, besides, search systems index it better.

Tables are applied for representation of table data only

During block layout, tables are used of course, but only when they are required, for example, to display numbers and other table data. An option of refusing tables at all is unwise, moreover harmful.

Let's summarize. In HTML4 and XHTML, layer is a basic element of the web page layout, when styles are actively applied and HTML and CSS specifications are supported. Under such an approach, the `<div>` tag plays important role, which is often associated with layers. At some point, it is true, so from now forth let's agree to apply the term layer to the `<div>` tag for which style identifier or class is specified. Thus, the expression "layer with the content name" means that the `<div id="content">` or `<div class="content">` tag is used.

Several new markup tags are added to HTML5 to designate various typed block of a page. For instance, `<header>` and `<footer>` are used to create sections in the top margin and bottom margin of a document, `<nav>` is used for navigation, `<aside>` for side panel. Adding such elements to the HTML specification aimed at decrease of the `<div>` tag domination and giving more meaning to markup. That's why the term element is actively used in the HTML5 layout, it means a corresponding tag and element it creates.

The above block layout principles are preserved at that, except that `<div>` is replaced by more meaningful tags in some cases.

Block Layout

As a rule, web page consists of many various elements that can have a complex structure. That's why when creating a web page, there is a need to position these elements in the right way, to style them so they are arranged on a page in the right way. That is, an issue of page markup creation, its layout, appears.

There are various ways, strategies, and types of layout. Initially, table layout was widely used since tables allow splitting web page area into rows and columns very easily. Rows and columns are easy to manage, any content can be easily positioned in them. That is what has determined popularity of table layout.

But table layout does not create the most flexible pages design-wise, which is especially relevant in the world without a uniform screen resolution, instead it has large TV screens, small tablet and phablet screens, very small smartphone screens, etc. Table layout didn't manage to satisfy this variety of screens. So it was gradually replaced by block layout. Block layout is a relative name of layout methods and techniques, when the CSS's float property is mainly used for layout of web pages, and the main building element of web pages is `<div>`, i.e. block. Using the float property and div or other elements, you can create a page structure consisting of several columns like in table layout, but it is much more flexible.

The float property was discussed in one of the previous topics. Now we use it to create a two-column web page. Let's suppose that we have standard header and footer at the top and bottom, and two columns in the center: menu column or sidebar and main content column.

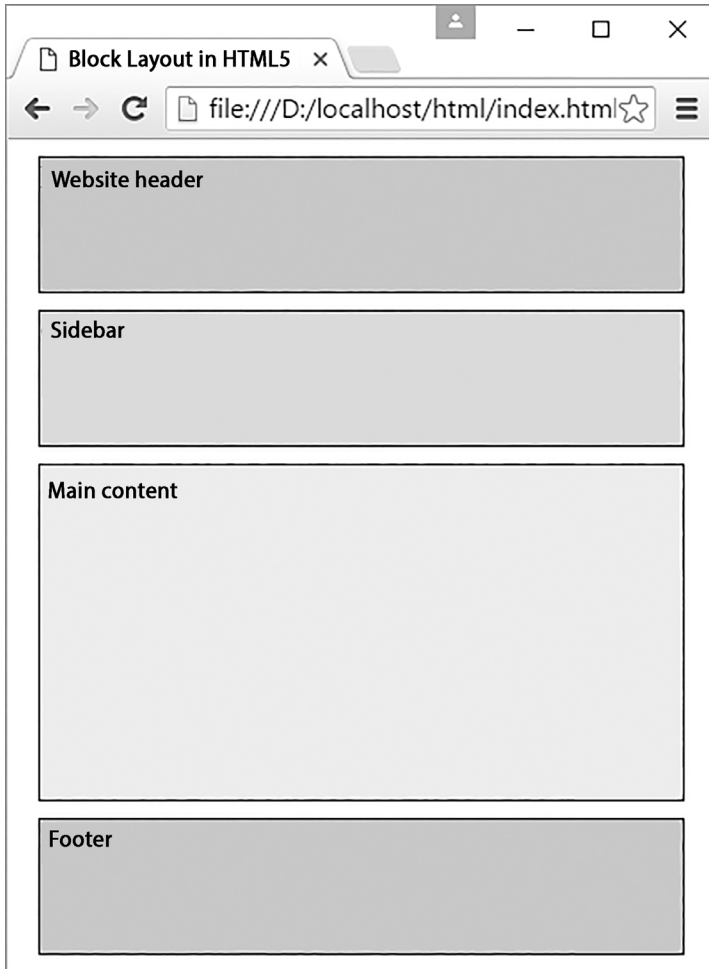
First we will specify all blocks. When working with elements that use wrapping and the float property, their order is important since the float element code that has the float property should go before the element that wraps the floating element. That is, the sidebar block will be stretched up to the main content block:

```
<!DOCTYPE html>
<html>

  <head>
    <meta charset="utf-8">
    <title>Block Layout in HTML5</title>
    <style>
      div{
        margin: 10px;
        border: 1px solid black;
        font-size: 20px;
        height: 80px;
      }
      #header{
        background-color: #ccc;
      }
      #sidebar{
        background-color: #ddd;
      }
      #main{
        background-color: #eee;
        height: 200px;
      }
      #footer{
        background-color: #ccc;
      }
    </style>
  </head>

  <body>
    <div id="header">Website header</div>
    <div id="sidebar">Sidebar</div>
    <div id="main">Main content</div>
    <div id="footer">Footer</div>
  </body>

</html>
```

Height, border, and block margins are added for decoration in this case, in order to identify the block area and separate it from others.

Then in order to move the sidebar block to the left relative to the main content block and to receive the wrapping effect, we need to specify the `float: left` property and preferred width of the sidebar block. Width may be fixed, for example,

150px or 8 em. Or you can also use percentage, for example, 30% — 30% of the body container width. On the one hand, fixed width blocks are easier to manage, but on the other hand, percentage value of width allows creating more flexible, responsive blocks that change their size relative to the change of the browser window size.

The last step is to specify margin between the main content block and sidebar block. Since the wrapping block can wrap a floating element on the right and at the bottom, if the floating element has a lower height, then we need to set a margin equal to the floating element width as a minimum. For example, if the sidebar width is 150px, we can set width of the main content block to 170px which allows creating blank space between two blocks.

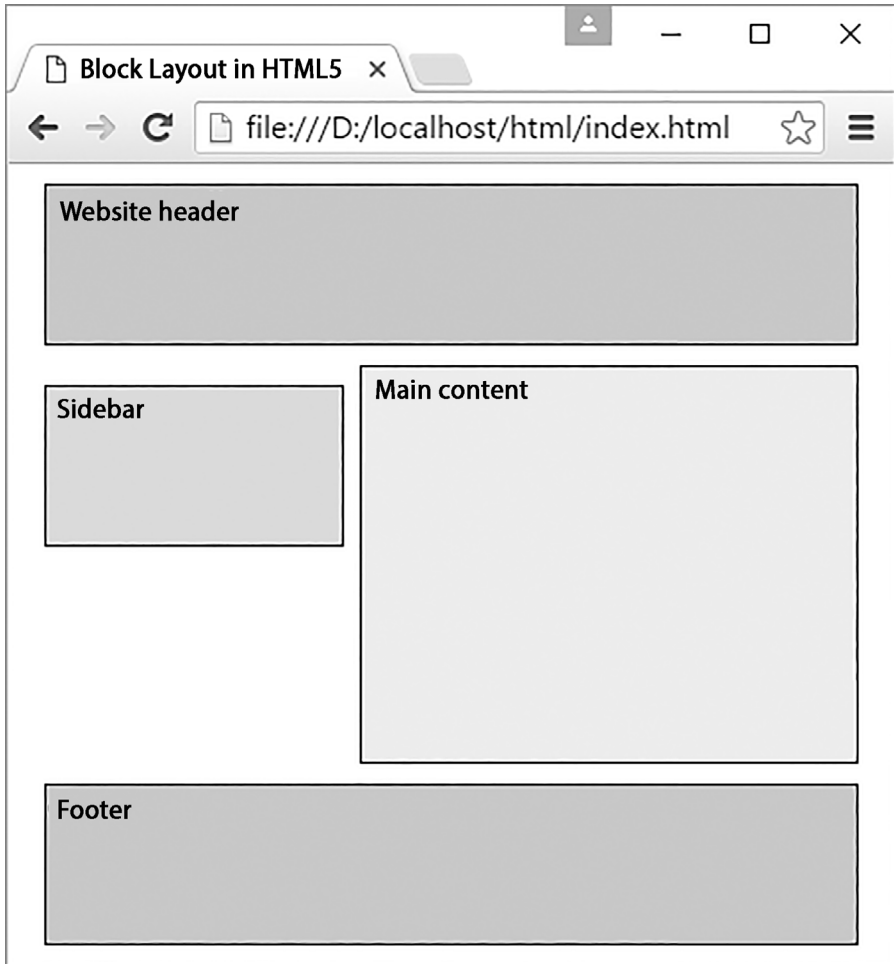
At that, it is not recommended to set width of the main content block explicitly since browsers widen it automatically to prevent it from occupying the whole space available.

So, taking into account everything above, let's alter styles of the sidebar blocks and main content as follows:

```
#sidebar{
    background-color: #ddd;
    float: left;
    width: 150px;
}

#main{
    background-color: #eee;
    height: 200px;
    margin-left: 170px; /* 150px (sidebar width) +
                        10px + 10px (2 margins) */
}
```

As the result, we get a sidebar to the left of the main block:

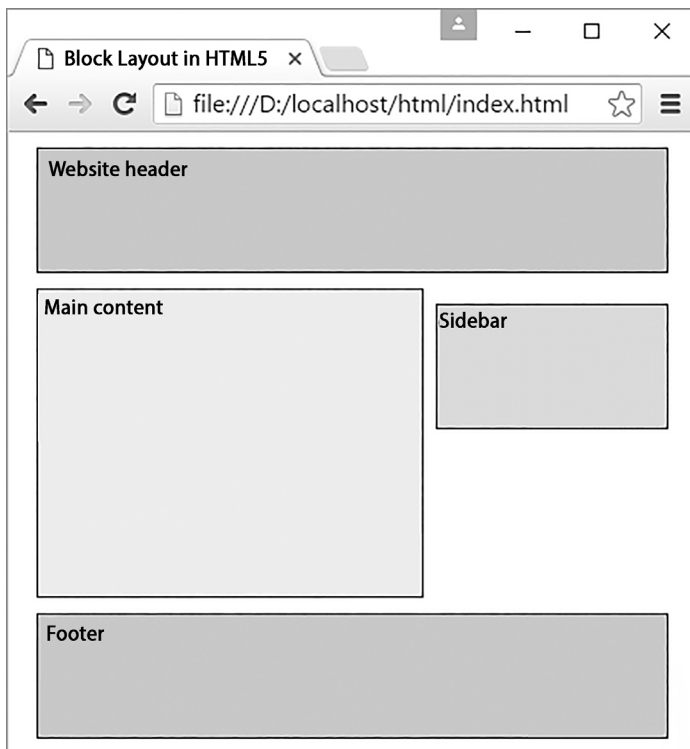


Block height is specified arbitrarily for better clarity; in real life, height is automatically specified by the browser as a rule.

Creation of the right sidebar is identical to creation of the left one, we just need to set the float: right value for the sidebar, and margin right for the main content block:

```
#sidebar{  
    background-color: #ddd;  
    float: right;  
    width: 150px;  
}  
  
#main{  
    background-color: #eee;  
    height: 200px;  
    margin-right: 170px;  
}
```

At that, the html markup is the same: sidebar block still must precede the main content block.

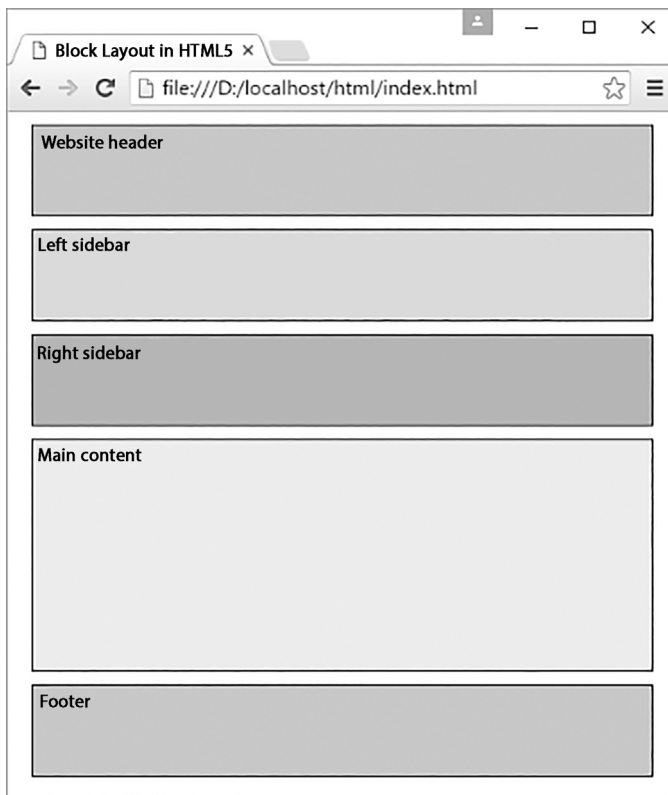


Block Layout: Part 2

The previous topic covered creation of a page with two columns. We can add a larger amount of columns in such a way and create a more complicated structure. For example, let's add a second sidebar to the web page:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Block Layout in HTML5</title>
    <style>
      div{
        margin: 10px;
        border: 1px solid black;
        font-size: 20px;
        height: 80px;
      }
      #header{
        background-color: #ccc;
      }
      #leftSidebar{
        background-color: #ddd;
      }
      #rightSidebar{
        background-color: #bbb;
      }
      #main{
        background-color: #eee;
        height: 200px;
      }
      #footer{
        background-color: #ccc;
      }
    </style>
  </head>
```

```
<body>
  <div id="header">Header of the web site</div>
  <div id="leftSidebar">Left sidebar</div>
  <div id="rightSidebar">Right sidebar</div>
  <div id="main">Main content</div>
  <div id="footer">Footer</div>
</body>
</html>
```



Here again, the code of both sidebars should go up to the main content block that wraps them.

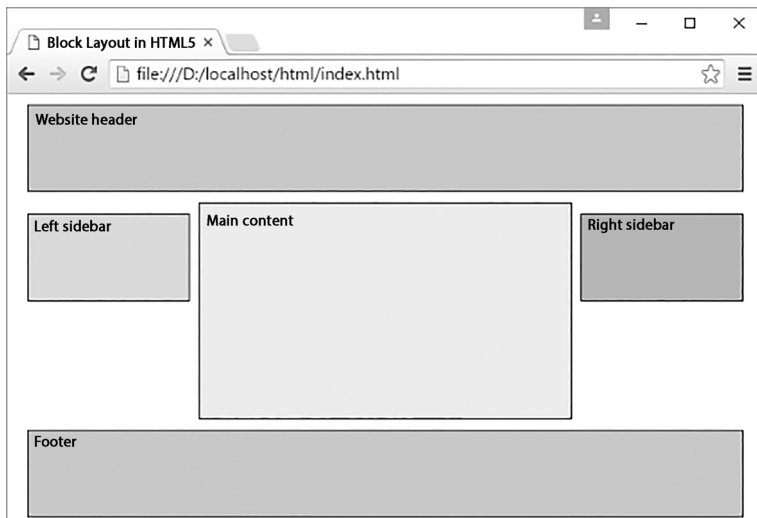
Now let's change styles of both sidebars and main content:

```
#leftSidebar{
    background-color: #ddd;
    float: left;
    width: 150px;
}

#rightSidebar{
    background-color: #bbb;
    float: right;
    width: 150px;
}

#main{
    background-color: #eee;
    height: 200px;
    margin-left: 170px;
    margin-right: 170px;
}
```

Again, we should specify width and the float property for both floating blocks (sidebars) one of them has the left value, and the other right.



Positioning of elements

Positioning is the position of the element in the coordinate system. There are four types of positioning: normal, absolute, fixed and relative. Depending on the type that is set via the position property, the coordinate system also changes.

Thanks to the combination of the position, left, top, right and bottom properties, the element can be superimposed on one another, output at a point with certain coordinates, fixed in a specified location, as well as we can determine the position of one element relative to another, and so on. Like other CSS properties, positioning control is available through scripts. In this way, you can dynamically change the position of the elements without reloading the page, creating animations and various effects.

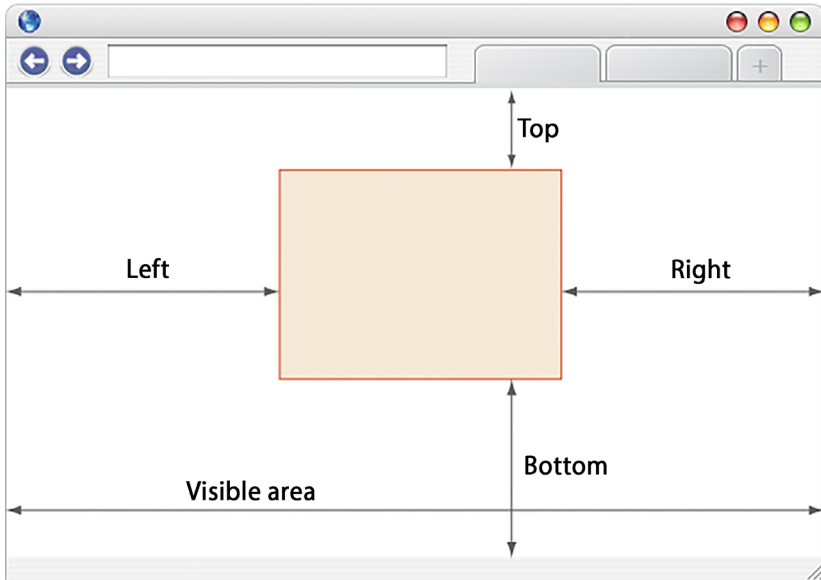
Normal positioning

If the position property is not specified for an element or its value is static, the element is displayed in the document flow as usual. In other words, the elements are displayed on the page in the order that they appear in the HTML source code.

The left, top, right, bottom properties are ignored if defined.

Absolute positioning

With absolute positioning, the element does not exist in the document flow and its position is set relative to the edges of the browser. You can specify this type via the absolute value of the position property. The coordinates are indicated relative to the edges of the browser window called the “visible area”



The mode is characterized by the following features.

- The width of the layer, if it is not explicitly specified, is equal to the width of the content plus the values of the margins, borders, and indents.
- The layer does not change its original position if it does not have the right, left, top and bottom properties.
- The left and top properties have a higher priority than the right and bottom. If left and right contradict each other, the right value is ignored. The same goes for bottom.
- If left is set to a negative value, the layer will go behind the left edge of the browser, the scroll bars will not appear. This is one way to hide an item from viewing. The same applies to the top property, only the layer will go beyond the upper edge.

- If left is set to a value greater than the width of the visible area or if to specify the right with a negative value, a horizontal scroll bar appears. A similar rule works with top, but it's a vertical scroll bar.
- Simultaneously, the specified left and right properties form the width of the layer, but only if width is not specified. But if you add the width property then the right value is ignored. Likewise, this happens with the height of the layer, only the top, bottom, and height properties are already involved.
- An element with absolute positioning moves along with the document when it is scrolled.

You can use the position property with the absolute value to create a frame effect. In addition to absolute positioning for elements, you must assign an overflow property with the auto value. Then when the content exceeds the height of the visible area, a scroll bar appears. The height and width of the "frames" is automatically formed by simultaneously using the left, right properties for the width, and top, bottom for the height

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

<head>
  <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
  <title>Absolute Positioning</title>
  <style type="text/css">
    body { margin: 0; }
    #sidebar, #content { position: absolute; }
```

```

#sidebar, #content {
    overflow: auto;
    padding: 10px;
}
#header {
    height: 80px; /* Layer height */
    background: #FEDFC0;
    border-bottom: 2px solid #7B5427;
}
#header h1 {
    padding: 20px;
    margin: 0; }
#sidebar {
    width: 150px;
    background: #ECF5E4;
    border-right: 1px solid #231F20;
    top: 82px; /* Distance from the top edge */
    bottom: 0; /* Distance from the bottom */
}
#content {
    top: 82px; /* Distance from the top edge */
    left: 170px; /* Distance from the left edge */
    bottom: 0; right: 0;
}
</style>
</head>
<body>
    <div id="header">
        <h1>Pilaf of the World</h1>
    </div>

    <div id="sidebar">
        <p>Fergana pilaf</p><p>Uzbek pilaf</p>
        <p>Siberian pilaf</p><p>Italian pilaf</p>
        <p>Estonian pilaf</p><p>American pilaf</p>
        <p>Native American pilaf</p>
    </div>

```

```

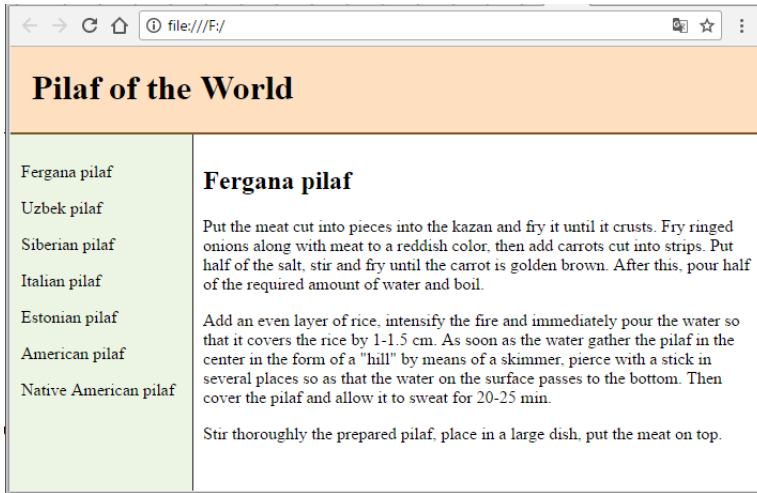
<div id="content">
  <h2>Fergana pilaf</h2>
  <p>Put the meat cut into pieces into
    the kazan and fry it until
    it crusts. Fry ringed onions along with meat
    to a reddish color, then add carrots cut
    into strips. Put half of the salt,
    stir and fry until the carrot
    is golden brown. After this, pour half
    of the required amount of water and boil.
  </p>
  <p>
    Add an even layer of rice, intensify
    the fire and immediately pour the water
    so that it covers the rice by 1-1.5 cm.
    As soon as the water gather the pilaf
    in the center in the form of a "hill"
    by means of a skimmer, pierce with a stick
    in several places so as that the water on
    the surface passes to the bottom. Then
    cover the pilaf and allow it to sweat for
    20-25 min.
  </p>
  <p>
    Stir thoroughly the prepared pilaf, place
    in a large dish, put the meat on top.
  </p>
</div>

</body>

</html>

```

The result of this example is shown. The header layer is output in the stream as usual, and absolute positioning is set for the sidebar and content layers.



In the IE6 browser, for absolutely positioned elements, you cannot set the left, right, top, and bottom properties at the same time.

Absolute positioning is also used to create various effects, for example, a pop-up tip to photos. Unlike the title attribute of the `` tag, which also displays the tip text, you can control the view of the text displayed using the script through the styles.

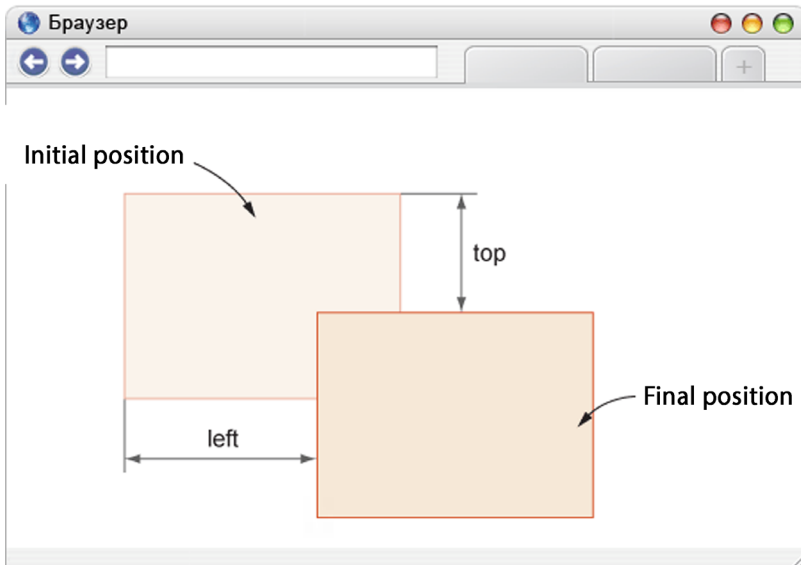
First, create an empty layer with the `floatTip` identifier and define its style. Three style properties are mandatory — position with the absolute value, display with the none value hides the layer and width sets the width of the layer with a tip. The remaining properties are used at the request of the developer and are designed to change the layer design

```
#floatTip {
    position: absolute; /* Absolute positioning */
    width: 250px; /* Block width */
    display: none; /* Hide */
    border: 1px solid #000; /* Border parameters */
}
```

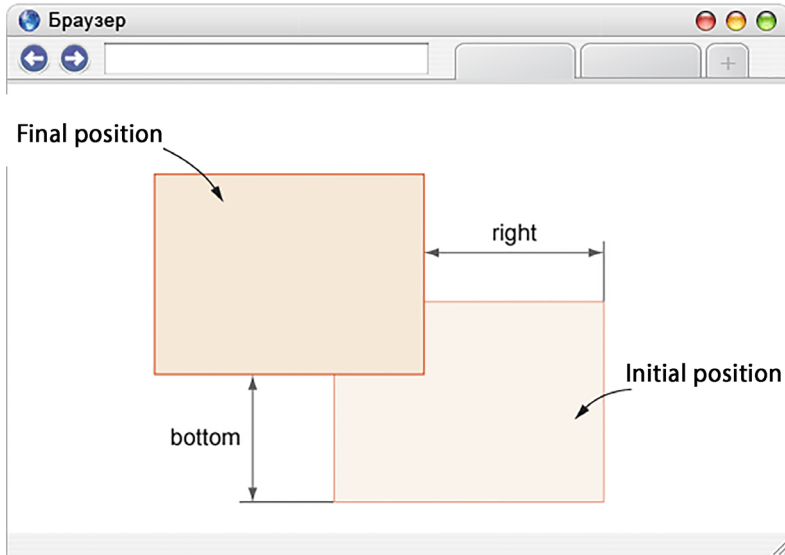
```
padding: 5px; /* Margins around the text */  
font-family: sans-serif; /* Sans-serif font */  
font-size: 9pt; /* Font size */  
color: #333; /* Text color */  
background: #ECF5E4; /* Background color */  
}
```

Relative positioning

If you set the relative value of the position property, the position of the element is relative to its original location. Adding the left, top, right and bottom properties changes the position of the element and shifts it to either side of the original location. The left positive value determines the shift to the right of the left border of the element, the negative value shifts to the left. A positive value of top sets the element shift down (Figure 3.46), negative — shift up.



The `bottom` and `right` properties produce the opposite effect. With a positive value, `right` shifts the element to the left of its right edge, with the negative value — shifts to the right (Figure 3.47). With a positive value of `bottom`, the element rises up, with a negative value — it goes down.



Relative positioning has the following features.

- This type of positioning is not applicable to table elements like cells, rows, columns, etc.
- When the element is displaced from the original position, the space occupied by the element remains empty and is not filled with below or above elements.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
```

```

<title>Title</title>

<style type="text/css">
    H1 {
        font: bold 2em Arial, Tahome, sans-serif;
        /* Font parameters */
        color: #fff; background: #375D4C;
        padding: 0 10px;
    }
    H1 SPAN {
        position: relative;
        /* Relative positioning */
        top: 0.3em; /* Shift down */
    }
</style>
</head>

<body>
    <h1><span>Font science from A to Z</span></h1>
    <p> Font is a means of expressing design,
        not some trivial reading.
    </p>
</body>

</html>

```

The z-index Property

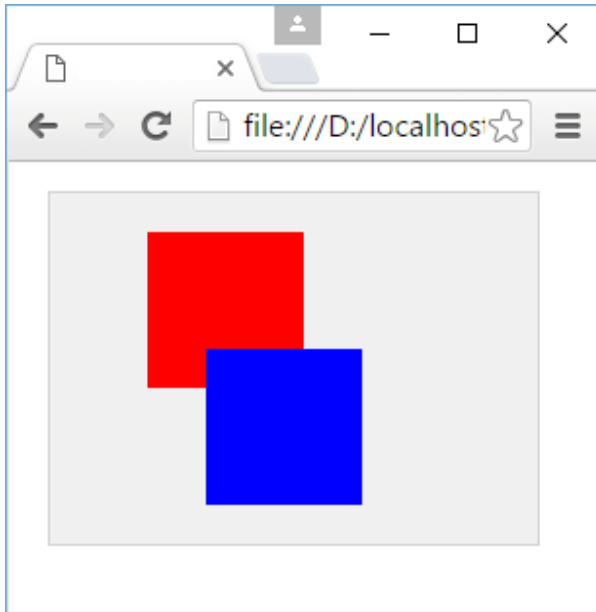
When borders of two elements coincide, the element determined last in html is displayed over another one by default. But the z-index property allows changing sequence order of elements when overlapping. The property takes number as a value. Elements with higher value of this property will be displayed over elements of the lesser z-index value.

For example:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Positioning in HTML5</title>
    <style>
      body{
        margin:0;
        padding:0;
      }
      .content{
        position: relative;
        top: 15px;
        left: 20px;
        width: 250px;
        height: 180px;
        background-color: #eee;
        border: 1px solid #ccc;
      }
      .redBlock{
        position: absolute;
        top: 20px;
        left:50px;
        width: 80px;
        height: 80px;
        background-color: red;
      }
      .blueBlock{
        position: absolute;
        top: 80px;
        left: 80px;
        width: 80px;
        height: 80px;
        background-color: blue;
      }
    </style>
  </head>
```

```
<body>
  <div class="content">
    <div class="redBlock"></div>
    <div class="blueBlock"></div>
  </div>
</body>
</html>
```

Now let's add a new rule to the redBlock style:



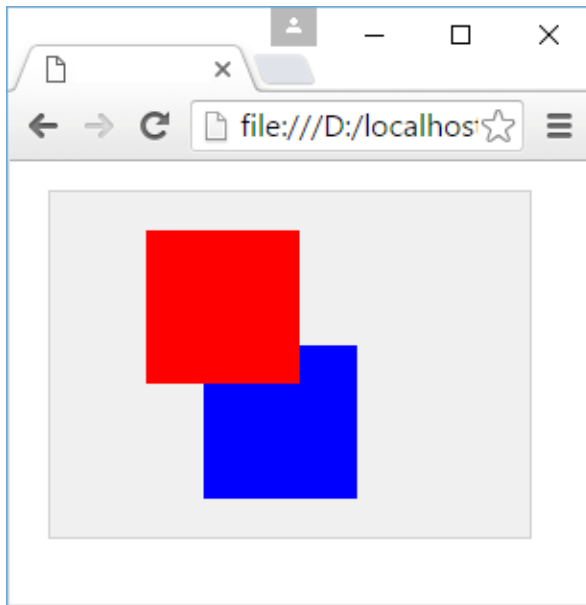
```
.redBlock{
  z-index: 100;
  position: absolute;
  top: 20px;
  left: 50px;
  width: 80px;
```

```
height: 80px;  
background-color: red;  
}
```

Here, z-index equals 100. But it doesn't have to be number 100. Since z-index is not defined in the second block and as a matter of fact it equals zero, we can set the z-index property to any value greater than zero for redBlock.

And now the first block will overlap with the second one, not vice versa as it was in the beginning:

Positioning in HTML5





Unit 8.

Web page block layout. Positioning.

© STEP IT Academy.

www.itstep.org

All rights to protected pictures, audio, and video belong to their authors or legal owners.

Fragments of works are used exclusively in illustration purposes to the extent justified by the purpose as part of an educational process and for educational purposes in accordance with Article 1273 Sec. 4 of the Civil Code of the Russian Federation and Articles 21 and 23 of the Law of Ukraine "On Copyright and Related Rights". The extent and method of cited works are in conformity with the standards, do not conflict with a normal exploitation of the work, and do not prejudice the legitimate interests of the authors and rightholders. Cited fragments of works can be replaced with alternative, non-protected analogs, and as such correspond the criteria of fair use.

All rights reserved. Any reproduction, in whole or in part, is prohibited. Agreement of the use of works and their fragments is carried out with the authors and other right owners. Materials from this document can be used only with resource link.

Liability for unauthorized copying and commercial use of materials is defined according to the current legislation of Ukraine.