

Web Pages and Web Sites

World Wide Web (www)

A set of documents (*web pages*) or files, which contain links to other documents/files and are published on the Internet

Web Browser

A program, which displays web pages (Firefox, Opera, Internet Explorer, Google Chrome, Safari)

Hypertext

A hypertext document is one that includes links (connections) to other documents - the links are usually distinguished by a different colour and are activated by the click.

Hyperlink

A link in the hypertext

URL = Uniform Resource Locator

The scheme for creating addresses in the World Wide Web

<i>Simplified URL Structure</i>			
http://	en.wikipedia.org	wiki	Earth
Protocol Identifier (what it is and how to process it)	Domain name (the place, where the page or file resides)	Path (in which folder/folders)	File (name of the object)
<i>ftp:// https://</i>	www.google.com	<i>/mobile/products/</i>	<i>mail.htm</i>

HTTP

HyperText Transfer Protocol - rules about how to transfer web pages

Website

A collection of web pages with the same domain name

Web Page

A document on the World Wide Web – it can contain text, pictures, movies, sounds, or links to other web pages

Web Server

A computer that stores and delivers web pages

Home Page

A web page, which is automatically displayed, when the domain name is entered in a browser (typically *index.html*, *index.php* or *index.aspx*)

Domain Name

(Internet) A name that identifies one or more IP addresses (e.g. a web server with a website)

DNS

A system, which translates domain names into IP addresses (e.g. www.google.com into 209.85.135.103)

HTML

- **H**yper-**T**ext **M**arkup **L**anguage
- an artificial language used to define the web page structure – i.e. it defines the *purpose/function* of certain web page elements (paragraphs, headers, lists, links...)
- originally it also helped to set up the web page design – nowadays replaced by CSS (cascading style sheets), which allow simple change of the page look (good examples: <http://www.csszengarden.com> – there is really the same HTML web page code with different CSS)
- current versions:
 - HTML 4.01
 - HTML 5 – the latest version
 - XHTML – XML-like HTML; designed for better interoperability with other data formats (database files, text processor files, spreadsheets, ...)

HTML Elements

- Two types
 1. **Container elements**
 2. **Empty elements**
- Indicated by tags – predefined symbols of the certain content types (*informally: they tell a browser what kind of content is about to be processed – if it is a paragraph, picture, link ...*), which are in the *angle brackets* (i.e. less than and greater than symbols).

Container Elements

- They have three parts
 1. **Start tag** (a symbol in angle brackets)
 2. **Content with the function defined by the start tag**
 3. **End tag** (the same symbol like a start tag with a slash)

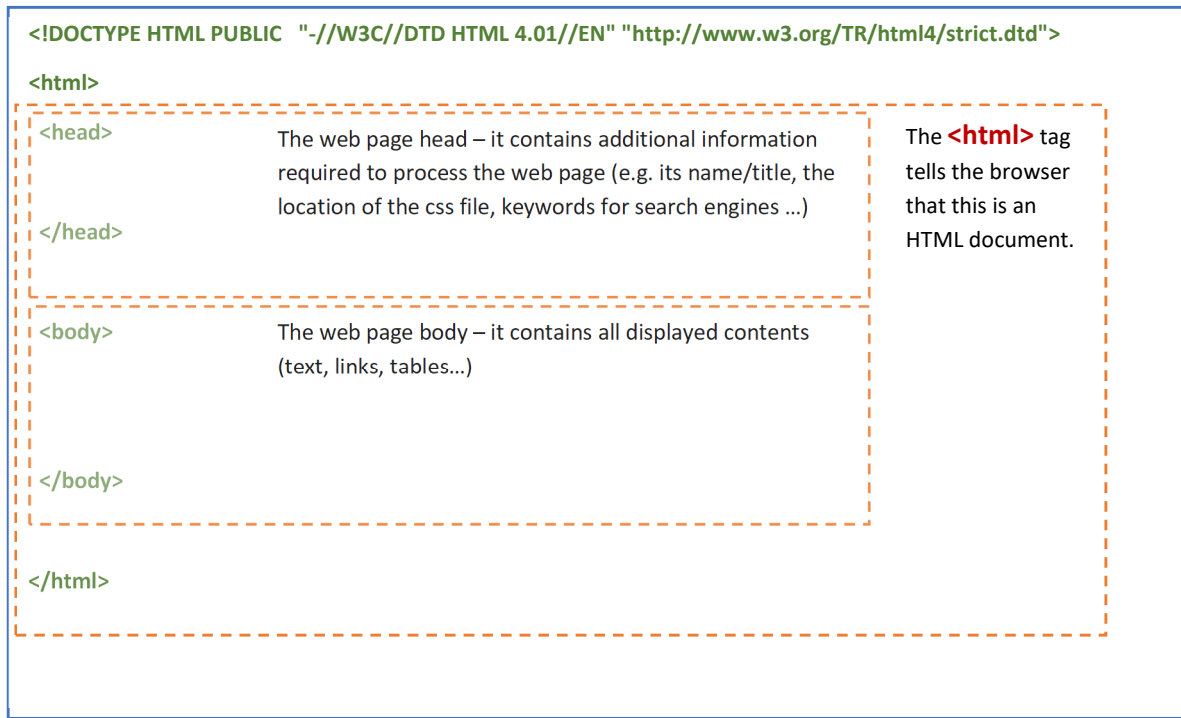
<p> *The p tag defines a paragraph of a web page.* **</p>**

Empty Elements

- Only a single tag, which inserts a **single** object (picture, line break) or pass an additional piece of information about the web page (name and location of the CSS file)

**
**

Web Page Structure



HTML Tags

`<p> ... </p>`

- It defines a paragraph – a continuous piece of text, which is automatically separated by some space before and after.

`<h1> ... </h1>`

- It defines a header – level 1, which should be used for the most important header/chapter title
- For subordinate levels – similar tags numbered from 2 to 6 (e.g. `<h2> ... </h2>` or `<h5> ... </h5>`)

` ... `

- It defines a list item – a part of the web page content introduced by a common symbol (unordered list) or an enumerator (a number/letter)

` ... `

- It defines an unordered/bulleted list
- Its individual items should be declared by the `` tags

` ... `

- It defines an ordered/numbered list
- Its individual items should be declared by the `` tags

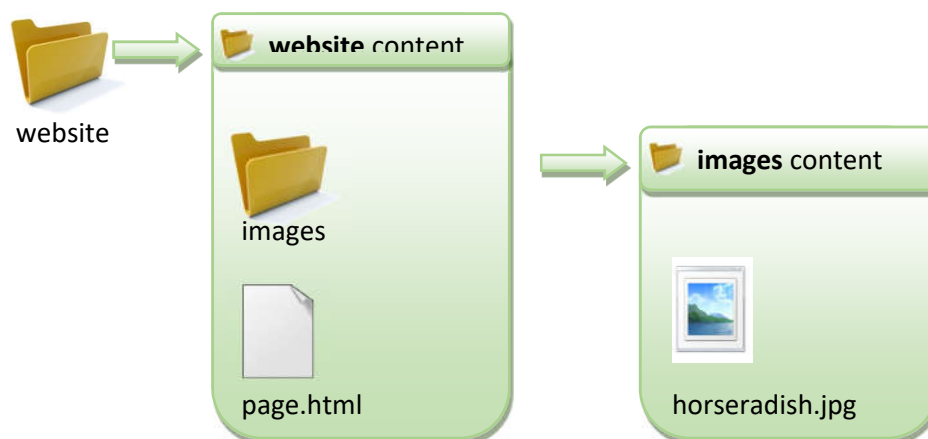
<a> ...

- It defines a link
 - to another page/file using **href** attribute
(e.g. `Open the Gmail!`)
 - to create a bookmark inside a document, by using the **name** attribute
(to see and try how it works, visit http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_link_bookmark)

- An empty element
- It inserts an image/picture into the web page
- There are required two attributes at least
 - **src** – source of the picture, i.e. its location and name
 - **alt** – alternative text – a description of the image, which is displayed, if the picture cannot be shown; also it is read by screen readers (blind people)
- Example

``

It shows the picture *horseradish.jpg*, which is stored in the subfolder of the location, where the page (e.g. *page.html*) with this code is located.



Tables

- Tables are defined with the `<table>` tag. A table is divided into rows (with the `<tr>` tag), and each row is divided into data cells (with the `<td>` tag). The letters td stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.¹

<table> ... </table>

- It defines one table – anything between the tags is considered to be the table content.

<tr> ... </tr>

- It defines one table row

<td> ... </td>

- It defines one cell of a row
- In case the cell spans several positions

¹ http://www.w3schools.com/html/html_tables.asp

- **colspan** – if the cell spans two or more columns
(e.g. `<td colspan="3">A long cell</td>` - the cell spans three columns)

	A long cell		

- **rowspan** – if the cell spans two or more rows
(e.g. `<td rowspan="2">A big cell</td>` - the cell spans two rows)

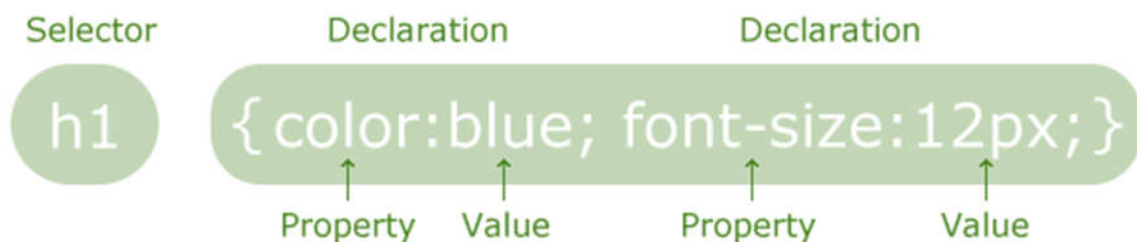
A big cell			

CSS – Cascading Style Sheets

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**
- Nice demo: http://www.w3schools.com/css/demo_default.htm

CSS Syntax

A CSS rule has two main parts: a selector, and one or more declarations:



The selector is normally the HTML element you want to style.

Each declaration consists of a **property** and a **value**.

The property is the style attribute you want to change. Each property has a value.

CSS declarations always end with a semicolon, and declaration groups are surrounded by curly brackets:

```
p {color:red;text-align:center}
```

To make the CSS more readable, you can put one declaration on each line, like this:

```
p
{
  color:red;
  text-align:center;
}
```

Text in **each** paragraph of the page is red.

Text in **each** paragraph of the page is centered.

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers.

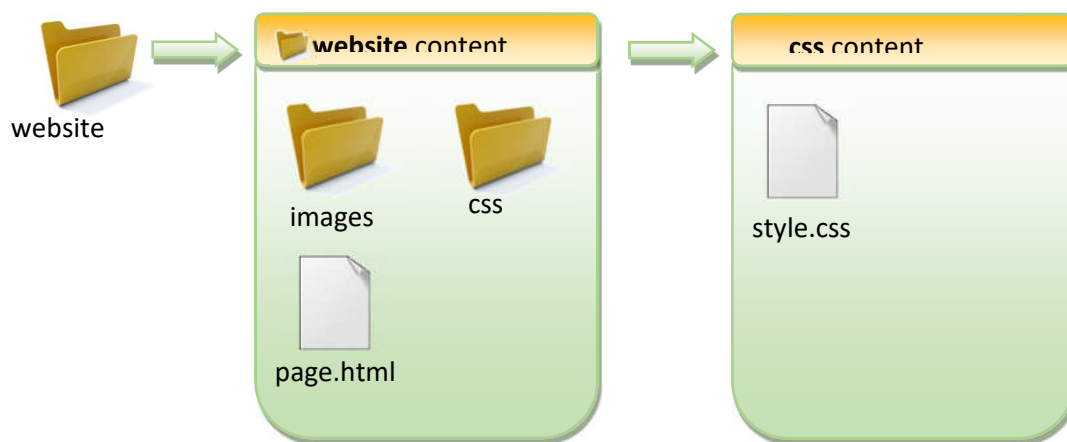
A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
p
{
  text-align:center;
  color:black;
  font-family:arial
}
```

/*This is another comment*/

CSS and HTML

There are several ways how to combine HTML code with CSS. In our case we will use an external CSS file linked into a web page.



In the example above the *page.html* should contain in its head the following code:

```
<head>
  <link rel="stylesheet" type="text/css" href="css/style.css" />
</head>
```

CSS Background

Property: **background-color**

- It sets the background of the chosen element
- The color can be defined by
 - its name (e.g. red, blue, fuchsia, cyan – the whole list is available at http://www.w3schools.com/css/css_colornames.asp),

- **RGB mix** – there is set the intensity of the red(R), green(G), and blue (B) color (e.g. rgb(255,255,0) is yellow),
- **Hex value** – another form of the RGB (the rgb(255,255,0) is #FFFF00 in hex) – there are tools, which determine the hex code for a given color – an online instance is the [ColorSchemeDesigner](http://colorschemedesigner.com/) (<http://colorschemedesigner.com/>)

```
p {
  background-color: green;
}
h1 {
  background-color: #de349e;
}
```

Property: **background-image**

- It applies a picture as the element background.
- By default, the image is repeated so it covers the entire element.

```
body {
  background-image: url('images/back.jpg') ;
}
h1 {
  background-image: url('images/flower.png');
}
```

Property: **background-repeat**

- The background-repeat property sets if/how a background image is repeated
- By default, the image is repeated so it covers the entire element.
- Possible values:
 - repeat-x – the picture is repeated *horizontally* only
 - repeat-y – the picture is repeated *vertically* only
 - no-repeat – the picture is displayed once
 - repeat – the default behavior

```
body {
  background-image: url('images/back.jpg');
  background-repeat: repeat-x;
}
h1 {
  background-image: url('images/flower.png');
  background-repeat: no-repeat;
}
```

Property: **background-position**

- The background-position determines where the background picture is aligned to. There are typically two values – the first for vertical position and the second for the horizontal one.
- Possible values:
 - top, center, bottom – vertical direction
 - left, center, right – horizontal direction
- There is also a way how to setup exact position (see http://www.w3schools.com/css/pr_background-position.asp)

```
body {
  background-image: url('images/back.jpg');
  background-repeat: repeat-x;
  background-position: top left;
}
h1 {
  background-image:
  url('images/flower.png');
  background-repeat: no-repeat;
  background-position: bottom center;
}
```

Margin and Padding

All HTML elements can be visually separated from other elements using the **padding** and **margin**. Padding creates space, which – among other things – gets the background of the element (e.g. if a paragraph background is black, then the padded area is black as well). Margin gets the background of the superior element (e.g. if the paragraph is inside a div, then margin gets its background).

Example:

```
h1 {
  padding: 10px;
  margin: 10px 40px;
}
```

Both can be defined using:

- 1 number – the same distance for all directions (e.g. **padding: 10px;**)
- 2 numbers – the 1st sets distance above+below, the 2nd on sides
- 4 numbers – individual distances in this order (like clock): top, right, bottom, left (e.g. **padding: 10px 20px 30px 40px;**).

// from this point on you may know what is here, but you do not have to

CSS positioning

Source: <http://www.barelyfitz.com/screencast/html-training/css/positioning/>

position:static

The default positioning for all elements is *position:static*, which means the element is not positioned and occurs where it normally would in the document.

Normally you wouldn't specify this unless you needed to override a positioning that had been previously set.

```
#div-1 {  
    position: static;  
}
```

Example

CSS	Result
<pre>p, div#example { width: 400px; margin: 0; } p {width: 300px;} #div-before, #div-after { background-color: #88D; color: #000; } #div-1 { background-color: #000; color: #FFF; padding: 1em; } #div-1-padding { padding: 10px; } #div-1a { background-color: #D33; color: #FFF; } #div-1b { background-color: #3D3; color: #FFF; } #div-1c { background-color: #33D; color: #FFF; }</pre>	 <p>id = div-before</p> <p>id = div-1</p> <p>id = div-1a</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam Phasellus ultrices viverra velit.</p> <p>id = div-1b</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam Phasellus ultrices viverra velit. Nam mattis, arcu ut bibendum commodo, magna nisi tincidunt tortor, quis accumsan augue ipsum id lorem.</p> <p>id = div-1c</p> <p>id = div-after</p>

position:relative

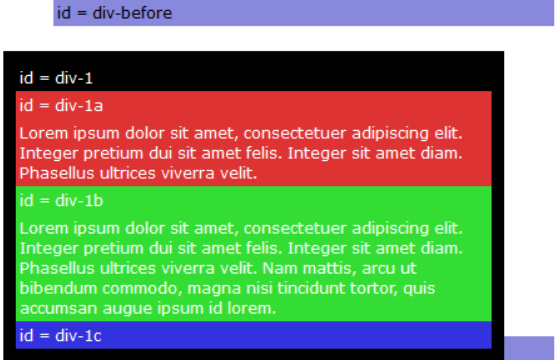
If you specify *position:relative*, then you can use *top* or *bottom*, and *left* or *right* to move the element relative to where it would normally occur in the document.

Let's move div-1 down 20 pixels, and to the left 40 pixels:

```
#div-1 {  
  position: relative;  
  top: 20px;  
  left: -40px;  
}
```

Notice the space where div-1 normally would have been if we had not moved it: now it is an empty space. The next element (div-after) did not move when we moved div-1. That's because div-1 still occupies that original space in the document, even though we have moved it.

It appears that *position:relative* is not very useful, but it will perform an important task later in this tutorial.

CSS	Result
<pre>p, div#example { width: 400px; margin: 0; } p {width: 300px;} #div-before, #div-after { background-color: #88D; color: #000; } #div-1 { background-color: #000; color: #FFF; position: relative; top: 20px; left: -40px; } #div-1-padding { padding: 10px; } #div-1a { background-color: #D33; color: #FFF; } #div-1b { background-color: #3D3; color: #FFF; } #div-1c { background-color: #33D; color: #FFF; }</pre>	

position:absolute

When you specify *position:absolute*, the element is removed from the document and placed exactly where you tell it to go.

Let's move div-1a to the top right of the page:

```
#div-1a {  
  position:absolute;  
  top:0;  
  right:0;  
  width:200px;  
  
}
```

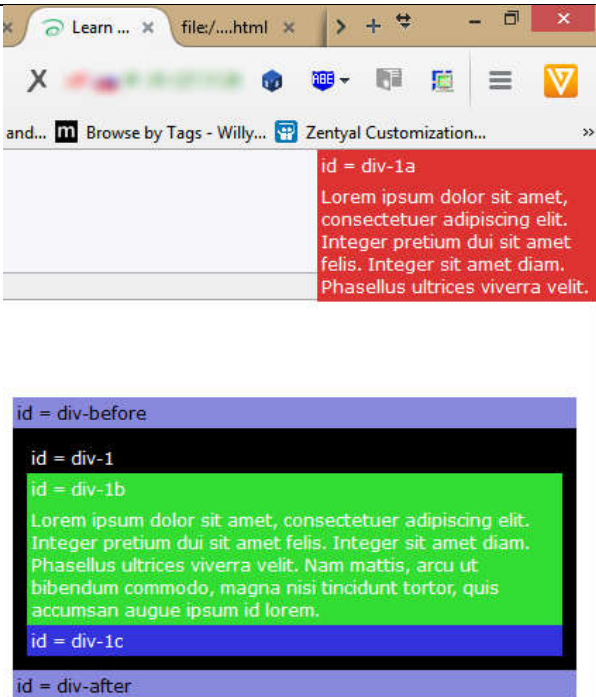
Notice that this time, since div-1a was removed from the document, the other elements on the page were positioned differently: div-1b, div-1c, and div-after moved up since div-1a was no longer there.

Also notice that div-1a was positioned in the top right corner of the page. It's nice to be able to position things directly the page, but it's of limited value.

What I really want is to position div-1a *relative* to div-1. And that's where relative position comes back into play.

Footnotes

- There is a bug in the Windows IE browser: if you specify a relative width (like "width:50%") then the width will be based on the parent element instead of on the positioning element.

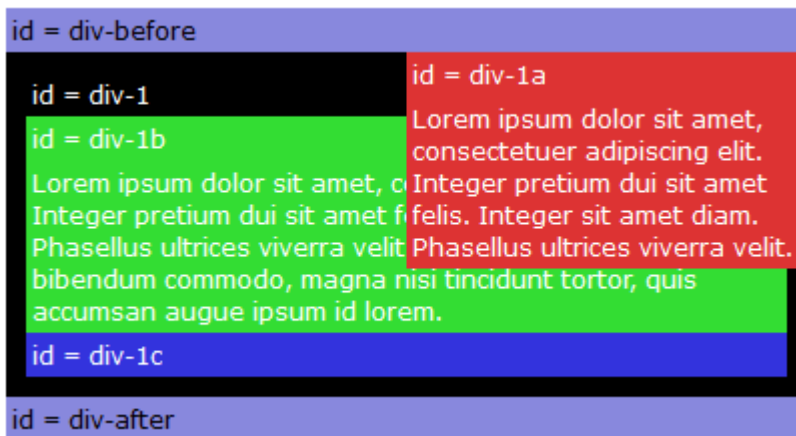
CSS	Result
like in the previous example with this change: <pre>#div-1a { position:absolute; top:0; right:0; width:200px; }</pre>	

position:relative + position:absolute

If we set *relative* positioning on div-1, any elements within div-1 will be positioned relative to div-1. Then if we set absolute positioning on div-1a, we can move it to the top right of div-1:

```
#div-1 {  
  position:relative;  
}  
  
#div-1a {  
  position:absolute;  
  top:0;  
  right:0;  
  width:200px;  
}
```

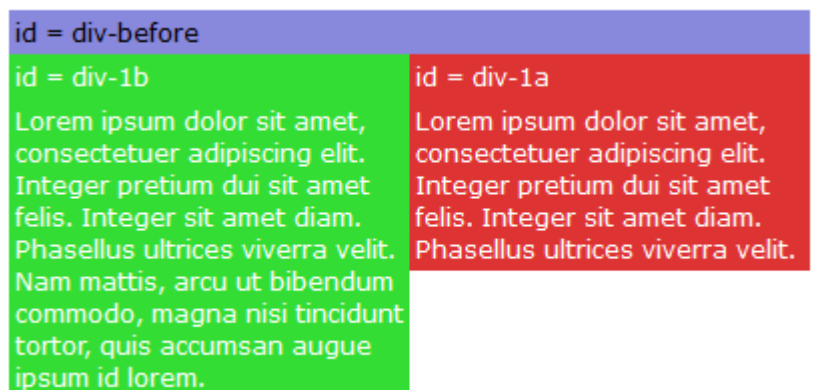
Result



two column absolute

Now we can make a two-column layout using relative and absolute positioning!

```
#div-1a {  
  position:absolute;  
  top:0;  
  right:0;  
  width:200px;  
}  
  
#div-1b {  
  position:absolute;  
  top:0;  
  left:0;  
  width:200px;  
}
```



two column absolute height

One solution is to set a fixed height on the elements.

But that is not a viable solution for most designs, because we usually do not know how much text will be in the elements, or the exact font sizes that will be used.

```
#div-1 {  
  position: relative;  
  height: 250px;  
}  
#div-1a {  
  position: absolute;  
  top: 0;  
  right: 0;  
  width: 200px;  
}  
#div-1b {  
  position: absolute;  
  top: 0;  
  left: 0;  
  width: 200px;  
}
```

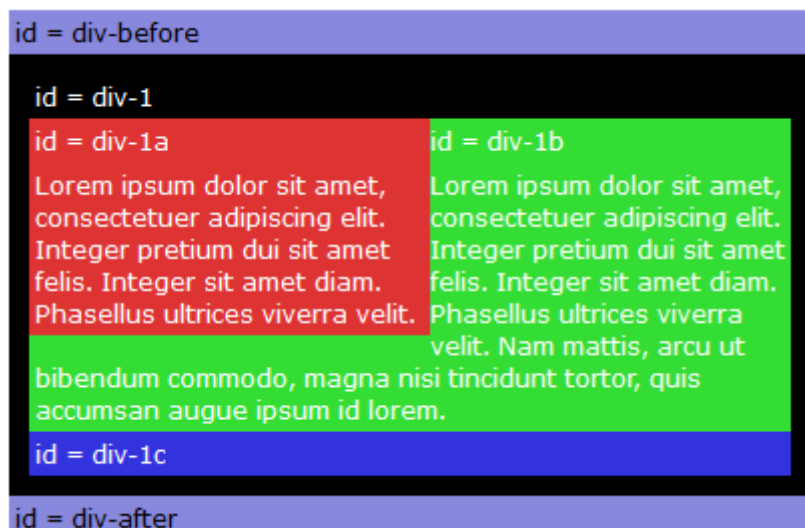


float

For variable height columns, absolute positioning does not work, so let's come up with another solution.

We can "float" an element to push it as far as possible to the right or to the left, and allow text to wrap around it. This is typically used for images, but we will use it for more complex layout tasks (because it's the only tool we have).

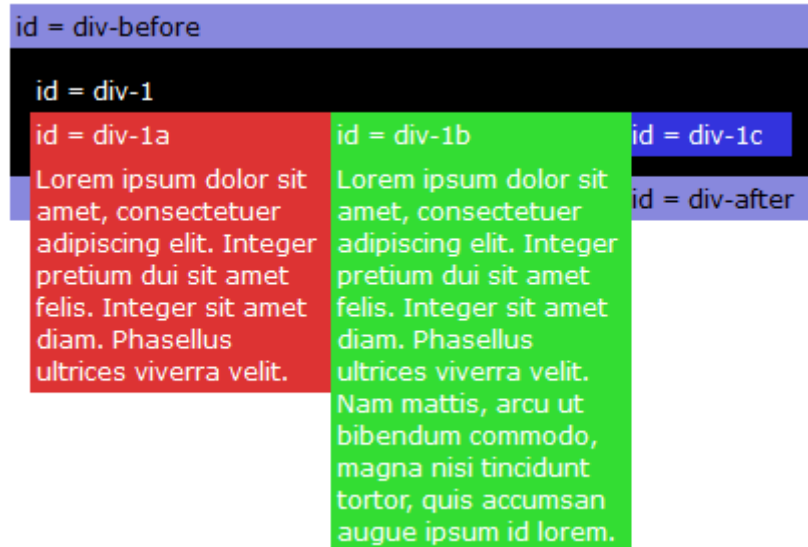
```
#div-1a {  
  float: left;  
  width: 200px;  
}
```



float columns

If we float one column to the left, then also float the second column to the left, they will push up against each other.

```
#div-1a {  
  float:left;  
  width:150px;  
}  
#div-1b {  
  float:left;  
  width:150px;  
}
```

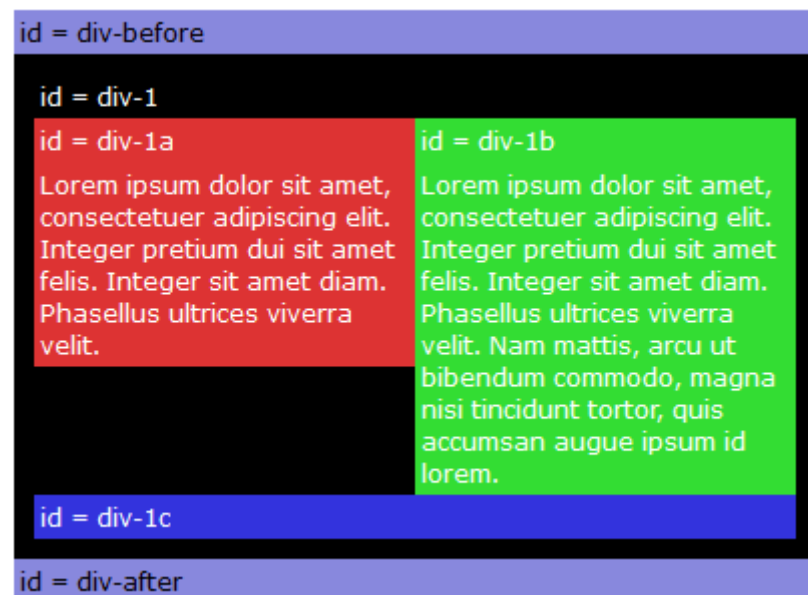


float columns with clear

Then after the floating elements we can "clear" the floats to push down the rest of the content.

```
#div-1a {  
  float:left;  
  width:190px;  
}  
#div-1b {  
  float:left;  
  width:190px;  
}  
#div-1c {  
  clear:both;  
}
```

These examples are extremely simplified and do not trigger some of the CSS bugs in the Windows IE browser (of which there are *many*).



display

`display` is CSS's most important property for controlling layout. Every element has a default display value depending on what type of element it is. The default for most elements is usually `block` or `inline`. A block element is often called a block-level element. An inline element is always just called an inline element.

block

`div` is the standard block-level element. A block-level element starts on a new line and stretches out to the left and right as far as it can. Other common block-level elements are `p` and `form`, and new in HTML5 are `header`, `footer`, `section`, and more.

inline

`span` is the standard inline element. An inline element can wrap some text inside a paragraph ` like this ` without disrupting the flow of that paragraph. The `a` element is the most common inline element, since you use them for links.

none

Another common display value is `none`. Some specialized elements such as `script` use this as their default. It is commonly used with JavaScript to hide and show elements without really deleting and recreating them.

This is different from `visibility`. Setting `display` to `none` will render the page as though the element does not exist. `visibility: hidden;` will hide the element, but the element will still take up the space it would if it was fully visible.

other display values

There are plenty of more exotic display values, such as `list-item`, `table`, `inline-block`, and `flex`.