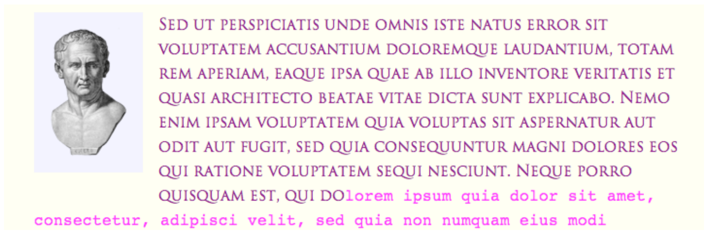
CSS 102

In this session we will expand on the styling and layout that we began in the previous example shown below. We took a very plain simple document and added fonts and layout to it.



SED UT PERSPICIATIS UNDE OMNIS ISTE NATUS ERROR SIT VOLUPTATEM ACCUSANTIUM DOLOREMQUE LAUDANTIUM, TOTAM REM APERIAM, EAQUE IPSA QUAE AB ILLO INVENTORE VERITATIS ET QUASI ARCHITECTO BEATAE VITAE DICTA SUNT EXPLICABO. NEMO ENIM IPSAM VOLUPTATEM QUIA VOLUPTAS SIT ASPERNATUR AUT ODIT AUT FUGIT, SED QUIA CONSEQUUNTUR MAGNI DOLORES EOS QUI RATIONE VOLUPTATEM SEQUI NESCIUNT. NEQUE PORRO QUISQUAM EST, QUI DO`lorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi`

This evening I will explain how those styling rules work.

REVIEW OF CSS 101

- Why CSS – to provide a consistent system for styling HTML documents
- What is CSS – Cascading Style Sheets
    - Styles in the document or in a separate file
    - Code like **`h1{font-size:24px;}`**
    - The cascade – later styles override earlier ones
    - Styling rules **`selector {property:value;..}`**
    - Selectors: element, id, class and others
    - Specificity: making our selectors specific
    - Properties and values
- CSS versions
- Resources
- Demo using the Lorem Ipsum example.

STYLING TEXT

COLOUR

Text colour is set using the **`color`** rule. Note the American spelling of color and center in CSS. The **`color`** rule provides a variety of ways to specify a foreground colour for text:

- **`color:rgb(red, green, blue)`** red, green and blue values from 0 to 255
- **`color:rgba(red, green, blue, alpha)`** same as rgb, except that a transparency is provided by the fractional alpha value between 0 = transparent and 1=opaque.
- **`color:#rrggbb`** six character hexadecimal value, you can get these from the PhotoShop colour dialog or from a color picker. The values represent red, green and blue. Eg. Cyan is **`#00ffff`**, equivalent to **`rgb(0,255,255)`**
- named colours, names like **`yellow`**, **`green`**, **`lightskyblue`**. If you use a good editor it will provide help with these.
- **`color:transparent`** completely transparent, equivalent to **`rgba(0,0,0,0)`**

BACKGROUND COLOUR

The **`background-color`** rule defines the colour of the background for any HTML element including text. It uses the same values as the **`color`** rule.

FONT FAMILY

The font-family rule defines a list of font families to be used for text. This list is searched until a font family is found that can be rendered by the browser. A typical font family would be **`font-family: futura-pt, helvetica, arial, san-serif;`** This tells the browser to first try using the Adobe TypeKit font Futura(paratype), if it can't find that use Helvetica on a Mac, or Arial on a PC and if none of those work use any sans-serif font it has.

FONT SIZE

Font sizes can be specified in using keywords, absolute values such as points and picas or relative values such as ems and percent. Pixel is special, it's defined as relative since pixel size varies, but behaves like an absolute value. Some example font sizes are:

`font-size:100%` use the current size

`font-size:1em` use the current size `font-size:.5em` half the current size

`font-size:16px` 16 pixels high, this is the default size in most browsers

`font-size:1pc` one pica

`font-size:larger` bigger than the current size

FONT WEIGHT

Font weight controls whether the font is rendered heavier or lighter than normal. The wight can either be specified as a number that is a multiple of 100 or by name. Normal weight is 400, bold is 700. Some typical values are:

`font-weight:normal` neither light nor bold

`font-weight:bold` heavier than normal

`font-weight:100` very light

`font-weight:900` very heavy

FONT STYLE

Font style controls whether the text is normal or slanted. There are two kinds of slant, italic and oblique. Italic is usually a separate font face in the family, oblique is usually calculated by the browser. Typical examples are:

`font-style:normal` upright

`font-style:italic` use the italic face from the family

`font-style:oblique` slant the font uniformly

FONT

The font rule is a "shorthand" rule that lets you specify several font attributes in a single rule, eg:

`font:bold italic 32px helvetica, sans-serif` heavy italic sans

This is not recommended as more than one of the font properties allows the value to be set to normal and the size and family must be the last two values.

LEADING

Leading is specified using the `line-height` rule. Line height differs in that it includes the character as well as the leading. Examples:

`line-height: 1.5` the size of a line is 50% larger than the font

`line-height: 18px` the size of a line is 18px regardless of the size of the font

ALIGNMENT

Text can be aligned using the **`text-align`** and **`vertical-align`** rules. Eg:

`vertical-align:baseline` the usual alignment

`vertical-align:sub` a subscript

`text-align: center` center the text in its box

`text-align: right` align to the right

DECORATION

Text can be decorated with underlines and and other effects eg:

`text-decoration:none` remove decoration, often applied to <a>

`text-decoration:underline` underlined

`text-decoration:blink` flashing like its 1999

LAYOUT

INLINE VS BLOCK

HTML elements are displayed as either inline elements like text, or block elements like paragraphs. Inline elements are shown side by side in rows, block elements are stacked above each other. Most elements are block, the typical inline elements are images and spans. This property can be set using the display rule.

DISPLAY

Display can change the way an element is displayed, eg:

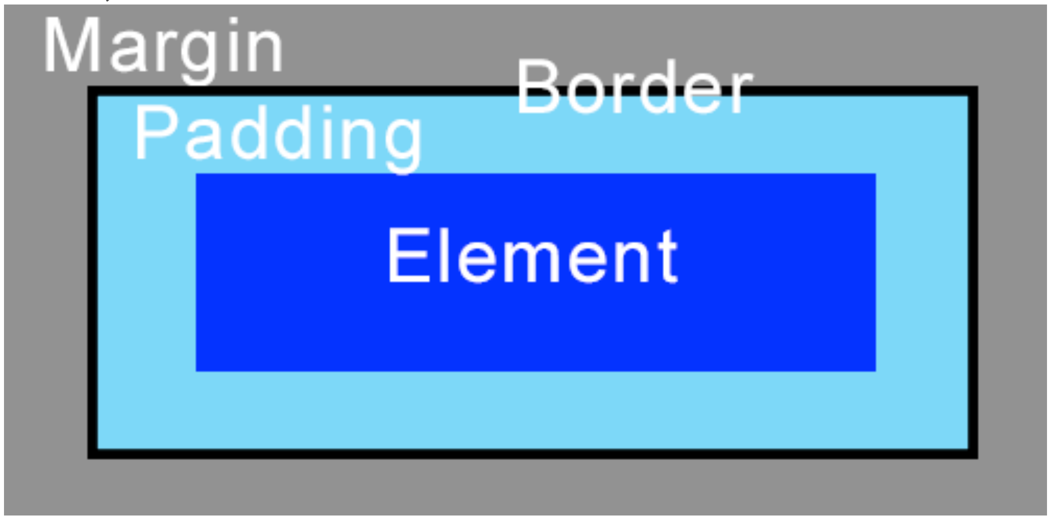
`display:block` display like a paragraph

`display:inline` display like text, cannot be used on an element that is normally a block

`display:inline-block` display a block inline

`display:none` do not display at all, this should not be used to hide text from readers for SEO purposes, Google won’t be happy about it.

PADDING, BORDERS AND MARGINS



Padding, borders and margins can be applied to most HTML elements. Padding is extra space around the element and takes on the background of the element. The border surrounds the padding and the margin separates the element from its neighbours. All of these can be specified individually for the top, right, bottom and left of the element or together in that order.

PADDING

The padding rule defines the amount of space immediately next to the element, eg:

`padding: .5em` half a em (character size) all around

`padding: .5em 0` half an em on the top and bottom, none on the sides

`padding: .5em 16px 0 1cm` half an em on the top, 16 pixels on the right, none on the bottom and 1 centimeter on the left

BORDERS

The border surrounds the padding and can be styled either using a shorthand rule or individual rules for the various sides and attribute eg:

`border:2px solid black` a narrow solid black border that applies to all sides, this is the typical way to specify a border

`border-top:5px solid red` a wide red border on the top only

`border-left-color:blue` blue colour for the left border

`border:none` no border, this is the default

MARGINS

Margins are space between elements or between an element and its container. The sizes are specified in the same way as padding, eg:

`margin: .5em 0` half an em on the top and bottom, none on the sides

`margin-left: 20px` leave 20 pixels between this element and the left side

`margin: 0 auto` automatic margins on the left and right, this is frequently used to center a block within its container. It only works if the block has a width

HEIGHT AND WIDTH

The height and width of a block is usually determined by its contents. This can be overridden using the **`height`** and **`width`** rules. These rules take the same sizes as we've seen on text, margins, etc. The value auto causes the size to be based on the content. Eg:

`width:100%` fill the container

`height:12em` 12 characters high

`width:auto` calculate the size based on content

POSITION

Blocks can be positioned either in the normal flow of the document, relative to an enclosing block, absolutely or on a fixed location on the screen. This is controlled by the position rule, eg:

`position:static` the default value, the element is placed in the normal flow

`position:relative` the element's position is relative to its container bur otherwise still in the normal flow of the document. Positioning rules such as left and top apply. This is often used to contain absolute positioned elements

`position:absolute` the element's position is relative to the nearest container that has positioning, it is outside the normal flow and doesn't occupy space in the document. This is often used for drop down menus, it is easy to abuse with DreamWeaver.

`position:fixed` the element's position is relative to the nearest container that has positioning, it is outside the normal flow and doesn't occupy space in the document. This is used for floating calls to action and menus.

FLOATING

Floating is used to display blocks side by side. A block can be floated to either the left or the right. The **`float`** and **`clear`** rules control this. This seems simple, but is actually very difficult to get right, often displaying blocks as inline-block works better. Floated content normally does not take up vertical space, so if you put a paragraph after it, the paragraph will wrap around it.

Some rules to control floating:

`float:left` float the block to the left, if another block has already been floated to the left this ends up just to the right of it

`float:right` float the block to the right

`clear:both` clear all floating in this box

`overflow:hidden` a kludge to get floated blocks to actually take up the vertical space that they appear to, you apply this style to the container

TABLES

Tables were once used for layout, they are now reserved for the display of tabular data. Occasionally they are used for layout, but only in very specialized situations.