CSS formatting: a style sheet with comments

/* Demonstration of style rules. I have used my own suggestions for colours and fonts etc. You can use your own choices of course. */

/* Style rules for html-tag "body": background colour and text colour (foreground colour). I style rule terminology calls the html tag for a "selector". */

```
body {
background-color: #ffcc99;
background-image: url(images/background.gif);
background-repeat: repeat;

/* Another possibilities here are repeat-y (vertical) and repeat-x (horizontally) or no-repeat (image is not repeated). You can also set the position for the background image: background-position: center; left or right. Positioning in more detail is also possible. */

/* Then it is what font-family and colour we want for the text. The selector is still the "body" tag. */

color: #000000;
font-size: 90%;
font-family: “Book Antiqua”, “Times New Roman”, serif;
}

/* If you have two and two characters the same in the colour value, then you can abbreviate to three characters. Example: #000000 for black can be written #000.*/

/* That was the body selector done! Maybe you just want to sett the background and text colour? You should always set rules for both foreground and background even if most browsers will show you black text on white background. Maybe you will leave it to the browser to decide font-family, font-color and font-size? It can be a good idea to use relative sizes for the text, em or % */

/* Lets set style rules for headings, use the html-tags h1, h2, h3, h4 as selectors. It is not often we use more than three levels of headings. The selectors are grouped because they shall have the same style rule for font-family .*/

h1, h2, h3, h4 {
font-family: Verdana, Helvetica, Arial, sans-serif;
}
h1, h2 {
color: #660000;  /* could have been #600. Short version could have been used in several places in this style sheet. */
}
h1 {
font-size: 180%;
}
h2 {
```

```
font-size: 150%;
}
h3 {
color: #cc3300;
font-size: 120%;
}
h4 {
color: #996600;
font-size: 100%;
}

/* That was headings done. */

/* Lets concentrate on links, unvisited links, visited links, hovering over a link (rollover effect) and active link (when clicking the mouse). */

a:link {
color: #3333cc;
text-decoration: none; /* Removes the underlining */
}
a:visited {
color: #6699cc;
text-decoration: none;
}
a:hover {
background-color: #ffcc99;
text-decoration: underline;
}
a.active {
color: #ff3300;
text-decoration: none;
}

/* Maybe you have a menu where you want the links to look different from links in the text. We can make a special class style for the special links in the menu. A class is made with a period and a name as selector. A class style can be used over and over again in the same file if we want to.*/

/*A bit special with pseudoclasses (links here). Several ways to do this. */

a.menu:link {
color: #660000;
text-decoration: none;
}
a.menu:visited {
color: #660000;
text-decoration: none;
}
a.menu:hover {
color: #660000;
```

```
text-decoration: underline;
}
a.menu:active {
color: #660000;
text-decoration: none;
}

/* Now the menu links are done, if you need them at all. In the html code it is necessary to
add the attribute ”class” and the value (name of the class) to each link you want to have the
formatting.
<a href= ”http://www.nb.no” class="menu">Menu item clickable</a> */

/* You can also make the class style for the links a bit more understandable perhaps. If we had
given our menu area the id value of menu <div id=”menu”>…. </div> then we would have to
use # instead of period .  */

.meny a:link {
color: #660000;
text-decoration: none;
}
.meny a:visited {
color: #660000;
text-decoration: none;
}
.meny a:hover {
color: #660000;
text-decoration: underline;
}
.meny a:active {
color: #660000;
text-decoration: none;
}

/* More on classes. Perhaps all your paragraphs, <p>, have a special look and you want one
<p> to be different. Make a class and apply it to the special <p>. If you want the class to
apply to only paragraphs and nothing else, then you may perhaps write the style rule with the
html tag first, then period and then the class name, for example: p.ingress in the css sheet and
in the html document it will be, as usual: <p class=”ingress”>…..</p>. */

/* If you think the style rule for your ingress will be used once only in your file (which is
likely), you have the option to use #ingress (an id instead of class). In the html document:
<p id=”ingress”>…..</p>. */

.ingress {
font-family: Georgia, "Times New Roman", Times, serif;
font-size:110%;
color:#333333;
background-color:#ffffff
}
```

/* About positioning of images. You want to position an image to the left and have the text flow around to the right of the image. Perhaps also with a bit of space between the image and the text. If you want some images to the left and some to the right, then you must use two classes as selectors, for example **.leftimage** and **.rightimage**, (up to you to find a class name but no Norwegian characters or spaces allowed) otherwise use the img tag as selector. */

```
img {
float: left;
margin-right: 0.8em;
margin-bottom: 0.8em;
}
```

/* About lists. Perhaps you want your lists to look a particular style different from the default rendering the browser gives you. Square bullet points or an image as a bullet point for bullet point lists and perhaps roman numerals for numbered lists. */

```
ul {
list-style-image: url(images/image.gif);
}
ol {
list-style-type: lower-roman;
}
```

/* About tables. Borders and padding can be used in many contexts. Here we use css to decide the formatting of the table grid. When we use the html element table as a selector and set a border, then the border will only apply to the outer edges of the table, not to the individual cells. When we set a border on every cell, then we get double borders. In order to avoid that, set a css rule on the table selector: border-collapse: collapse. */

```
table {
border-collapse: collapse;
width: 600px;   /* Use percent for the width for adaption to the browser window, 80%; */
margin-left: auto;  /* Centring in browser window with left and right margins set to auto */
margin-right: auto;
}
th, td {
border: black solid thin; /* Short version used, not border-color… etc */
padding: 0.2em;
}
th {
text-align: left;  /* Default is that content in header cells are bold and centred */
color: #ff0000;
font-weight: normal;
background-color: fef8dc;
}
td {
vertical-align: top;
}
caption {  /* Header which is part of the table.*/
font-size: 120%;
```

```
font-style: italic;
padding-bottom: 0.4em;
}
```

Measurement in CSS

Relative sizes:

- **em.** (The size of an M). If you set font-size 2em on a heading then the heading will be twice the size of the rest of the text. ”em” can be used in many situations and is recommended for font-, margin- and padding sizes.
- **Ex.** (The size of an X). Not used a lot.
- **percent:** Font-size 120% gives you a font size 20% bigger than the browser would normally render the text.

Absolute sizes:

- **in:** Inches (one inch is 2,54 cm)
- **cm:** Centimetres
- **mm:** Millimetres
- **pt:** Points (one point is 1/72 inch, 0,0352 cm)
- **pc:** Pica (one pica is 12 points, 1/6 inch, 0,423 cm)

Size relative to screen resolution:

- **px**: Pixels will function mostly as an absolute value. Pixels are used a lot on the web.

Comments (truths with modifications):

- All values with the exception of px are independent of screen resolution. 12pt for text is supposed to look the same size on a screen with 1024x768 resolution as one with 800x600.
- Standard (default) font-size in Netscape, Opera and Internet Explorer is 12pt, equivalent to approximately 16px.
- Absolute sizes are used preferably only when the ”output media” is known or if you want to fit the file to you printer or if you absolutely want to!