

# Cascading style sheets

Practical Guide for Business & Community Engagement





# Cascading Style Sheets

<b>Introduction .....</b>	<b>2</b>
<b>Task 1. Getting Started .....</b>	<b>3</b>
<b>Task 2. Using an Internal Style Sheet .....</b>	<b>5</b>
<b>Task 3. CSS for Colours and Fonts .....</b>	<b>7</b>
<b>Task 4. Internal or External CSS?.....</b>	<b>10</b>
<b>Task 5. Using Classes.....</b>	<b>12</b>
<b>Task 6. Pseudo-classes .....</b>	<b>13</b>

© Netskills, University of Newcastle

Copyright in the whole and every part of this Courseware whether in the form of a written manual, document, software program, service or otherwise belongs to the University of Newcastle upon Tyne ("the Owner") and may not be used, sold, licensed, transferred, copied or reproduced in whole or in part in any manner or form or in or on any media to any person other than in accordance with the terms of the Owner's Licence Agreement or otherwise without the prior written consent of the Owner.

All use of this material is governed by the Owner's Standard Licence Agreement together with the appropriate Schedule.

The following are available:

Standard Licence Schedule to cover all use including all for-profit use by any type of organisation and all use by non-educational establishments

Educational Licence Schedule for not-for-profit internal use only by a recognised educational establishment

The Netskills logo and this copyright notice must be included in any copy or adaptation.

Netskills is a trademark of Netskills, University of Newcastle

# Introduction

Business and Community Engagement (BCE) units across the education sector recognise that the good use of the World Wide Web can greatly enhance the effectiveness of their work. The key roll of providing: *an interface between universities and the rest of the world*, makes the use of the web crucial.

## Why Are Cascading Style Sheets Important?

It is expected that BCE web sites deliver professional-looking, accessible and cost-effective content. Couple this with the need to provide easily maintained web pages that meet the current publishing standards and the need to use CSS (Cascading Style Sheets) becomes essential when developing a web site.

CSS is a language used to describe the presentation of a document. It defines colours, fonts, layout, and other aspects of the document and provides for the separation of document content from the presentation.

## Who is it for?

This module is aimed at those broadly involved in Business and Community Engagement who are likely to be involved in the creation and development of web pages and web sites.

These areas of work might also be termed Third Stream, Third Strand, Third Mission, Employer and Community Engagement, Knowledge Exchange, Enterprise and Innovation or Knowledge Transfer.

You should be comfortable working with HTML code and understand the basic tags and attributes used to structure a web page.

## Useful resources that help with this module include:

### CSS Online Tutorial

[www.w3schools.com/css](http://www.w3schools.com/css)

### Why use CSS?

[www.westciv.com/style\\_master/academy/css\\_tutorial/introduction/key\\_ideas.html](http://www.westciv.com/style_master/academy/css_tutorial/introduction/key_ideas.html)

### Free CSS Templates

[www.freetemplatesonline.com](http://www.freetemplatesonline.com)

# Task 1. Getting Started

<b>Objectives</b>	To locate a web browser and text editor on your PC and to access a web site to obtain the sample files used in these tasks and download copies to edit locally.
<b>Method</b>	You will use a web browser to access the files before downloading them to your local file area.
<b>Comments</b>	The plain HTML documents you are about to download make up a small website that currently has no additional style or formatting information applied to it. Over this sequence of tasks you will be using CSS to add the style yourself!

---

**Important Note** If you have not already done so, it is a good idea to make a note of where you should be saving files on the computer you are working on. You should also create your own folder in order to keep your files separate.

If you already know where to save your files and have created a local folder to keep them in then go straight to Task 1.3.

**Task 1.1** Locate your main web browser (e.g. Internet Explorer) and a text editor (e.g. Windows Notepad). You will need both of these applications throughout these exercises.

If you are attending a Netskills workshop, your trainer should have told you how to access these applications on the PC you are using.

**Task 1.2** Create a new folder in your local file area called `basics`.

**Task 1.3** In your web browser go to the following location:

<http://materials.netskills.ac.uk/resources/42stylesheets/>

You should see the following list of files:

```
about.html
logo.gif
on-site.html
portfolio.html
tonic.html
```

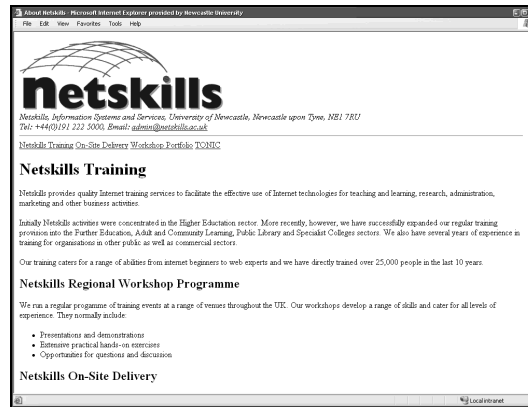
Using your mouse, *right-click* on the link to [about.html](#) and select `Save Target As...` from the pop-up menu (or `Save Link As...` if you are using Firefox or Netscape) and save the file to the `basics` folder in your local file area.

Repeat this for all the files in the list.

**Note** Make sure your files are saved with a `.html` extension (Internet Explorer has a nasty habit of attempting to change them to `.htm`).

**Task 1.4**

Open your local copy of `about.html` in your web browser. It should look similar to Figure 1-1 below.



**Figure 1-1 Plain `about.html`**

Take a moment to familiarise yourself with the layout and content in the page before using the links at the top of the page to browse to the other parts in the example site.



**Figure 1-2 Example site links**

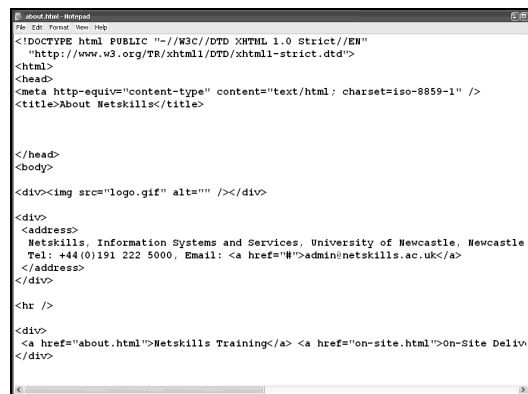
Notice how the pages are not formatted in any way but that they make sense when read and link together logically.

**Navigation Note**

Some of the links on these pages will take you away from the example site you will be working with. If you follow such a link you will get a page warning you that you are about to leave the example web site. Just use your browser's back button to return.

**Task 1.5**

When you have finished browsing the site, open `about.html` in your text editor and take a look at the source code it contains.



**Figure 1-3 `about.html`**

You should find that there are no specific formatting or layout instructions – just plain HTML tags and attributes.

This might not look like the most exciting set of web pages you have ever seen but they are clean, valid, accessible and perfectly useful. This is the framework upon which we can hang our CSS formatting – safe in the knowledge that without CSS the user will always get a useable, accessible, valid web page.

**Task 1.6**

Finally return to your text editor – You are now ready to create a style sheet!

**Important Note**

If you take look at the source code for the other HTML files you will notice that they appear to have a number of different tags and attributes that are not in `about.html`.

Don't panic! – You will be editing `about.html` to add some of this code yourself – we have just saved you the effort of having to add the same code to 4 files each time!

# Task 2. Using an Internal Style Sheet

- Objectives** To create an internal CSS style sheet to format an individual web page
- Method** You will use a text editor to create an internal CSS style sheet for about .html. You will also use your text editor to add style rules to the style sheet and observe the effect in a web browser
- Comments** An internal style sheet is the simplest way to apply CSS style to a web page. It is created in the head of the web page and the browser uses the rules it contains to format and display the page. Style rule specified in an internal style sheet only apply to the current page.

## Task 2.1

In your text editor, add the code shown in bold below to the <head> section of about .html.

```
<html>
<head>
<meta http-equiv="content-type"
  content="text/html; charset=iso-8859-1" />
<title>About Netskills</title>

<style type="text/css">
  body {background-color: #CCCCCC;}
</style>

</head>
```

Save your file and open it in your web browser – you should notice that the page background has changed colour (see Figure 2-1)



Figure 2-1 Background colour changed

- Help!** If nothing has changed in about .html, check the following things:
- Nothing changed!**
1. Have you saved about .html after editing it?
  2. If about .html was already open in your browser, did you refresh/reload it?
  3. Take a look at the code you have added – does it match the example above?  
In particular check:
    - The US spelling of color
    - The brackets (they should be curly!)
    - The colons and semi-colons

If you still can't see the effect, then ask a trainer for help – another pair of eyes often helps!

**Task 2.2**

Browse to the other pages in the site – they should have remained unchanged...

**What have I just done?**

You have just added an *internal* style sheet to the <head> section of about .html. By placing it in the <head> the browser has access to it as it processes the page.

Any CSS style rules in this internal style sheet can *only* be applied to about .html. This is why the rest of the site has not changed (yet).

The style sheet itself is delimited by a pair of HTML <style> tags – along with an attribute (type) to tell the browser the style sheet language that it can expect to find between them. The type attribute simply allows for the possibility of other style languages being used – not just CSS e.g.

```
<style type="text/css">

</style>
```

At this point the style sheet does not contain any CSS rules. The first CSS rule you added to your style sheet is shown in bold below:

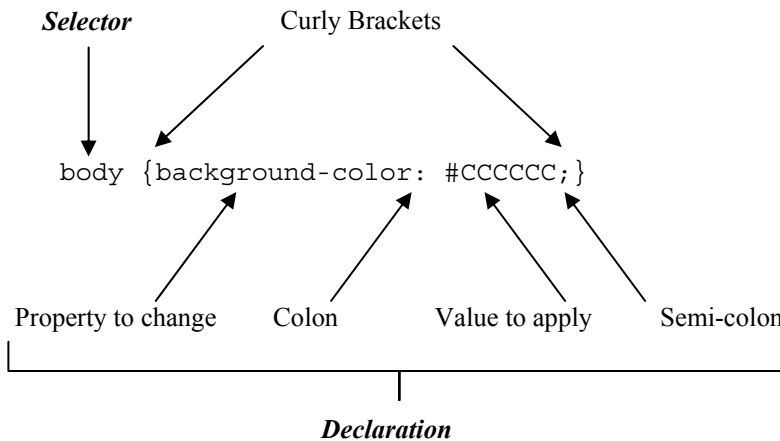
```
<style type="text/css">
  body {background-color: #CCCCCC;}
</style>
```

CSS style rules are made up of a *selector* – specifying what the browser should look for in the page and a pair of curly brackets containing one or more *declarations* – specifying what to apply when it finds it.

Each declaration consists of a *property* and the *value* you wish to set, separated by a colon (:). Multiple declarations are separated by a semi-colon (;).

As long as you get the punctuation correct (curly brackets, colons and semi-colons) white space and new lines between declarations is perfectly OK.

Notice the syntax for the rule you added:



Now you are ready to add some more declarations and start using CSS!

# Task 3. CSS for Colours and Fonts

<b>Objectives</b>	To add CSS declarations to specify colour and font information in a web page.
<b>Method</b>	You will use a text editor to add CSS declarations to an internal style sheet.
<b>Comments</b>	Colours and fonts are probably the CSS properties that you will set for virtually every page you create. They are also amongst the simplest things to specify with CSS and introduce you to a range of common bits of CSS syntax.

---

---

**Task 3.1** In your text editor specify a foreground colour for the body text as shown below.

```
<style type="text/css">
  body {background-color: #CCCCCC;
        color: #000000;}
</style>
```

Save your file and reload it in your browser.

**Nothing new happened... Why?** If you look at the page in your browser you will probably not notice any change. The text was black before and it is still black now. The difference is that your style sheet is now unambiguous – you have given the browser both a background *and* a foreground colour to use.

This is a very good habit to get into as the CSS and HTML specifications allow the browser to use it's own built in default styles or any locally specified user styles to display anything that is not explicitly declared in the CSS.

If you had not specified the foreground colour and a user had their text colour set to grey then they'd probably see a blank page and wonder what was wrong!

This doesn't mean that you have to think of everything that could possibly be formatted and apply a rule for it. The skill is to format clearly and unambiguously just what you need to be formatted. Hopefully you will develop some of that skill with these materials – the rest comes with practice and experience!

**Task 3.2** Add another declaration to specify the font-face the browser should use.

```
<style type="text/css">
  body {background-color: #CCCCCC;
        color: #000000;
        font-family: "Century Gothic", Arial, sans-serif;}
</style>
```

Save and reload about.html. You should see the font change for all the text in the page.

## **Netskills Training**

Netskills provides quality Internet training services teaching and learning, research, administration, ma

**Figure 3-1 Changing the display font**

**Why 3 fonts?** The declaration you have just added contains a list of fonts. The browsers can only use fonts that are available on the local machine. It will try them in order until it finds one it recognises. Fonts with spaces in their name should be quoted and the last one in the list should always be a generic font-family name. To the browser the declaration above reads  
*"Try Century Gothic...if that is not available, try Arial...failing that use any available sans-serif font"*

Common font-family values include:

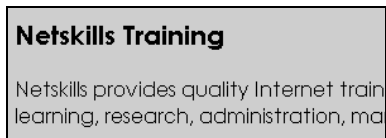
Verdana, Arial, Helvetica, sans-serif	Sans-serif types e.g. Some Text
"Times New Roman", Times, serif	Serif types e.g. Some Text
"Courier New", Courier, mono	Mono-spaced types e.g. Some Text

**Task 3.3**

Add the style rules for the heading tags in the page, as shown below.

```
<style type="text/css">
  body {background-color: #CCCCCC;
        color: #000000;
        font-family: "Century Gothic", Arial, sans-serif;}
  h1 {font-size: 130%;}
  h2 {font-size: 115%;}
  h3 {font-size: 105%;}
</style>
```

Save and reload about .html to observe the effect (the headings should all get smaller).



**Figure 3-2 Reduced headings**

**Common Font Properties and Units** CSS can be used to change all of the common font properties. You will see many of them in action throughout these materials.  
 Common font properties include:

font-size	The task above uses font-size. The units used are percentages (%). Other units can also be used and are either <i>absolute</i> or <i>relative</i> . (see following note).
font-weight	Usually given the value <b>bold</b> (other values include <b>normal</b> (which can be used to "unbold" headings) and <b>bolder</b> but can be buggy in some browsers).
font-style	Sets values for <i>italic</i> and <b>normal</b> (not italic!).

**Absolute or Relative Units?** Absolute units include points (pt) and pixels (px) and are fixed sizes, independent of the size of anything else in the page. They should generally be avoided for text as they do not allow text to be easily zoomed up in the browser – causing major accessibility problems for many users.

Absolute units start to come into their own when used for page design and layout, but that is the subject of another set of Netskills Materials!

Relative units are calculated in proportion to other items in the page and include percentages (%) – which are usually easy to get your head around – but also "ems" (em) and "exes" (ex) which calculate dimensions based on the size of either an m or an x in the relevant font.

So using relative units, the current text size for an element could be represented as 100% or 1em or 1ex. A doubling of the text size could therefore be declared using either 200% or 2em or 2ex.

Relative units should be used for text wherever possible, particularly as users can then use their browser to resize all the text in a page correctly. The relative unit you choose to does not really matter but it pays to be consistent!

A word of warning – if you don't specify units the browser will usually try and use pixels i.e. font-size: 8 will make the text 8 pixels high.

#### Task 3.4

Add the following rule to space out the text in all the headings by 0.1em

```
h1,h2,h3 {letter-spacing: 0.1em;}
```

Save about.html and return to your browser to see the effect.

**Multiple selectors** The last rule you added also introduces another key technique in CSS – the ability to apply a style rule using multiple selectors. In this case the letter spacing is being applied to <h1>, <h2> and <h3> tags, wherever they occur in the document.

**CSS Colours** The colours you have used in your style sheet have been specified using a *hex code*. These are the standard, unambiguous, way to provide colour values and represent the red/green/blue values in hexadecimal format – understood by all browsers and operating systems.

Other ways of specifying colours include:

**Hex** color: #ff0000

**Name** color: red

**RGB** color: rgb(255,0,0)

For more information see:

[http://www.w3schools.com/html/html\\_colors.asp](http://www.w3schools.com/html/html_colors.asp)

<http://www.w3.org/MarkUp/Guide/Style.html>

(scroll to the bottom of the page)

If you wish to experiment with different colour schemes try using a colour picker such as the one at:

<http://wellstyled.com/tools/colorscheme2/index-en.html>

# Task 4. Internal or External CSS?

- Objectives** To move the style rules declared in an internal CSS style sheet into an external file that can be used to format several pages.
- Method** You will open new file in your text editor and copy and paste the style rules created in previous tasks into it. You will then link the new external file to all the files in your sample web site.
- Comments** The style rules you have been creating up to this point are all contained in an internal style sheet and can only be applied to the web page they are declared in. External style sheets enable web authors to apply the same styles over an entire site. In reality most authors will use a combination of external and internal styles sheets to fully format a site.

## Task 4.1

In your web browser follow the links to the other pages of you web site. you should find that the CSS you have created in the previous tasks is only being applied to about .html

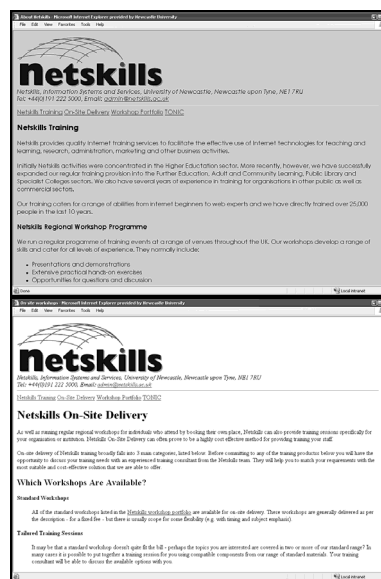


Figure 4-1 Internal CSS is only applied to one page!

## Task 4.2

In your text editor, return to about .html and use your mouse to highlight all the rules in the style sheet (shown in bold below). You do not need the <style> tags.

```
<style type="text/css">

  body {background-color: #CCCCCC;
        color: #000000;
        font-family: "Century Gothic", Arial, sans-serif;}

  h1 {font-size: 130%;}
  h2 {font-size: 115%;}
  h3 {font-size: 105%;}
  h1,h2,h3 {letter-spacing: 0.1em;}

</style>
```

Cut these from the page (usually Edit > Cut from the text editor toolbar).

## Task 4.3

Open a new, blank file in your text editor and paste in the style rules cut from about .html (usually Edit > Paste from the toolbar).

Save this file as **format.css**

**Task 4.4**

Return to `about.html` and find the (now empty) `<style>` tags.

```
<style type="text/css">
</style>
```

Replace these with the following line:

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

Save `about.html` again.

Refresh `about.html` in your browser – there should be no change in appearance!

**So what has changed?**

What you have done is to store all the style rules in a file called `format.css` and inserted a link to that file in `about.html`

When the browser loads `about.html` it can follow the link to `format.css` to retrieve the style information. This means that by inserting the same link in several pages, they can all use the same style sheet. Any changes to the style sheet only need to be made once – in `format.css` – saving you a lot of work should you wish to alter the appearance of your pages.

The important details in the `<link>` tag are as follows:

<code>rel="stylesheet"</code>	Specifies the relationship of the linked document to the current one i.e. it is a style sheet for the current page, not a link to be followed in its own right.
<code>type="text/css"</code>	Specifies the content type of the style sheet i.e. the language it uses. This caters for the possibility of browser supporting more than one style sheet format.
<code>href="format.css"</code>	Specifies the location of the style sheet. In this example it points to a relative location i.e. <code>format.css</code> is in the same directory location as <code>about.html</code>  It can also be a full URL, which is often better to avoid the problem of missing style sheets when pages are moved!

**Task 4.5**

Open `on-site.html` in your text editor and add the link from Task 4.4 to the `<head>` section i.e.

```
<html>
<head>
<meta http-equiv="content-type"
content="text/html; charset=iso-8859-1" />
<title>On-site workshops</title>
<link rel="stylesheet" type="text/css" href="format.css"/>

</head>
```

Save `on-site.html`. now repeat this for `portfolio.html` and `tonic.html`

**Task 4.6**

Finally return to your web browser and browse through the site.

You should find that they have all picked up the CSS rules from `format.css`

For the remainder of these tasks you will be adding CSS rules to `format.css` that will apply to *all* the pages in your sample site.

# Task 5. Using Classes

- Objectives** To introduce the use of CSS classes to apply different styles to groups of HTML elements.
- Method** You will use a text editor to add `class` attributes to an HTML document. You will then create corresponding CSS rules in a style sheet.
- Comments** Classes are a useful way of applying different style rules to subsets of the same element type. It is also possible to create generic classes that can be applied to any element.

---

**Task 5.1** Use your text editor to add a `class` attribute to the opening paragraph of `about.html` as shown below:

```
<p class="hi-lite">
Netskills provides quality Internet training services to
facilitate the effective use of Internet technologies for
teaching and learning, research, administration, marketing
and other business activities.
</p>
```

Save `about.html` and then add the following CSS rule to the bottom of `format.css`:

```
h1,h2,h3 {letter-spacing: 0.1em;}

/* div, span {border: solid thin yellow;} */

p.hi-lite {font-weight: bold;}
```

Save `format.css` and reload `about.html` in your browser to see that the first paragraph should now be rendered in bold.

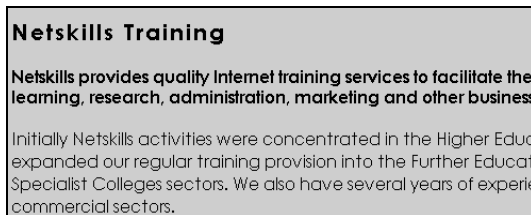


Figure 5-1 Applying the `p.hi-lite` rule

**Note** The rule you have just added **only** formats paragraphs that are **also** in the `hi-lite` class.

**Task 5.2** Now apply the attribute `class="hi-lite"` to all three `<span>` tags you added in **Error! Reference source not found.** e.g.

```
<p>
Our training caters for a range of abilities from internet
beginners to web experts and <span class="hi-lite">we have
directly trained over 25,000 people in the last 10
years</span>.
</p>
```

Save `about.html`

If you reload the page in your browser you should not see any change. This is because the CSS rule currently still only applies to paragraphs that are in the `hi-lite` class.

**Task 5.3** Convert the `hi-lite` rule in `format.css` into a generic class rule by removing the tag name from the selector (but leave the dot!):

```
.hi-lite {font-weight: bold;}
```

This now allows the browser to apply the rule to **any** tag that contains the attribute `class="hi-lite"`.

# Task 6. Pseudo-classes

- Objectives** To introduce the use of built-in pseudo-classes to alter the appearance of hyperlinks in a web page.
- Method** You will use a text editor to add pseudo-classes to a CSS style sheet
- Comments** Pseudo-classes are pre-defined as part of the CSS specification and can be used to apply CSS style to hyperlinks – more specifically to apply different styles for the different *states* of a link. You cannot define your own pseudo-classes.

---

**Task 6.1** Open `format.css` in your text editor and add the following rules to format the hyperlinks in the page:

```
#footer .smallprint {text-align: right;}

a:link {color: #0000ff;}
a:visited {color: #800080;}
a:hover {color: #ffffff;}
a:active {color: #ff0000;}
```

Save `format.css` and look at the effect in your browser.

Make sure you look, not only at what happens when you move your mouse over a link, but also at the effect produced by clicking a link once and holding down the mouse button.



**Figure 6-1** Moving over a link

Each of these actions changes the *state* of the link. Each of the pseudo-classes above relates to a different state and tells the browser how to make the link appear each time.

<code>a:link</code>	Normal appearance of a link in the page
<code>a:visited</code>	Appearance of a link that the user has visited
<code>a:hover</code>	Appearance of a link when the mouse passes over the link (no buttons pressed)
<code>a:active</code>	Appearance of a link in use – observed by clicking once and holding the mouse button down

Although you can't define your own pseudo classes you can use any CSS you like in the declarations used to format them. You can also use them as you would regular selectors (multiple lists, nested etc)

**Important Note** For reasons best known to Microsoft the pseudo-classes for links only work reliably if they are declared in the order shown in these tasks.

As this is not an issue in other browsers, it is a good idea to get into the habit of using them in the order used here.

# Appendix 1 format.css

At the end of these tasks you should have a series of web pages that all get their formatting instructions from a single external CSS style sheet.

The style sheet, called **format.css** has been built up in stages. If you have completed all the tasks `format.css` should look like this:

```
body {background-color: #CCCCCC;
      color: #000000;
      font-family: "Century Gothic", Verdana, Arial, sans-serif;}

h1 {font-size: 130%;}
h2 {font-size: 115%;}
h3 {font-size: 105%;}
h1,h2,h3 {letter-spacing: 0.1em;}

p.hi-lite {background-color: #666666; color: #cccccc;}

a:link {color: #0000ff}
a:visited {color: #800080;}
a:hover {color: #ffffff;}
a:active {color: #ff0000;}
```