

Advanced CSS

Steven Pemberton

CWI and W3C
Kruislaan 413
1098 SJ Amsterdam
The Netherlands

Steven.Pemberton@cwi.nl
www.cwi.nl/~steven

About the Instructor

Steven Pemberton is a researcher at the CWI, The Centre for Mathematics and Computer Science, a nationally-funded research centre in Amsterdam, The Netherlands, the first non-military Internet site in Europe.

Steven's research is in interaction, and how the underlying software architecture can support the user. At the end of the 80's he built a style-sheet based hypertext system called Views.

Steven has been involved with the World Wide Web since the beginning. He organised two workshops at the first World Wide Web Conference in 1994, chaired the first W3C Style Sheets workshop, and the first W3C Internationalisation workshop. He was a member of the CSS Working Group from its start, and is a long-time member (now chair) of the HTML Working Group, and co-chair of the XForms Working Group. He is co-author of (amongst other things) HTML 4, CSS, XHTML and XForms.

Steven was until recently Editor-in-Chief of ACM/interactions.

Objectives

HTML has for too long, and incorrectly, been seen as a purely presentation language. It was originally designed as a structure description language, but extra elements were added later by browser manufacturers in order to influence the presentation. This has had the effect of limiting Web site usability by introducing presentation elements that slow down Web access, reduce or prevent accessibility to the sight-impaired, limit the sorts of devices that may be used to view websites, and reduce the end-user's options when viewing a document.

The World Wide Web Consortium (W3C) started the Style Sheet activity in 1995 in order to get HTML back to its pure form. The result of this was Cascading Style Sheets (CSS), which allows the separation of content and presentation in Web sites.

Using style sheets has many benefits, including:

- Separation of content and presentation means that Web pages are easier to write.
- Since images are no longer needed to represent styled text, Web pages download significantly faster.
- By separating out the presentation elements, blind and other sight-impaired users are able to access the Web much more easily, especially since CSS explicitly supports aural browsers.
- By allowing style sheets to specify sizes in relation to other sizes, rather than as absolute sizes, people with reduced sight can scale pages up and still see them as they were intended.
- By not coding device-dependent presentations in the HTML pages are viewable on a wider range of devices.
- You can now design the look of your site in one place, so that if you change your house style, you only need to change one file to update your entire site.

Even if the Web remained based on HTML, these would be enough reasons to use style sheets. However, the Web is now going in a new direction: XML, and XML has no inherent presentation semantics at all. To use XML you *have* to use a style sheet to make your site visible.

As a part of the movement to XML, a new version of HTML, called XHTML, is being developed. Since all presentation-oriented elements are being dropped, style sheets will become essential there too.

So the objectives of this course are to give an advanced introduction to the use of CSS to style HTML and XML documents, covering in passing how this can improve usability.

Most attention will be paid to features introduced in CSS level 2

These notes have been produced entirely in XHTML and CSS, using different stylesheets for printing, screen use, and presentation.

It should go without saying that to properly appreciate this document, you have to view it with a browser with good CSS support.

1: Aims

The idea of this course is to introduce you to most features of CSS2, and illustrate how they can be used to good effect.

Along with a number of exercises, we will also be studying a number of advanced designs, with the question "How did they do *that*?"

This course assumes that you already know the basis of CSS

- How to apply it to a document
- The structure of CSS (Rules, selectors, properties, values)

Basic styling: fonts, colours, margins, borders, padding, text properties, background, and the basics of positioning.

However, it doesn't assume you have followed part one of the course, so some features mentioned there will be repeated.

2: About bugs

There are many different CSS user agents, and they are not all as good as each other.

While I will be mentioning some shortcomings in some browsers, this is a course on CSS, and not a loving compilation of all the bugs that there are to be found and how to design around them: a course on CSS should have lasting value; a course on which browser has which bugs wouldn't.

3: Motivation

Csszengarden.com is a website with essentially [one HTML page](#).

And hundreds of beautiful, breathtaking CSS stylesheets applied to that one page.

css Zen Garden

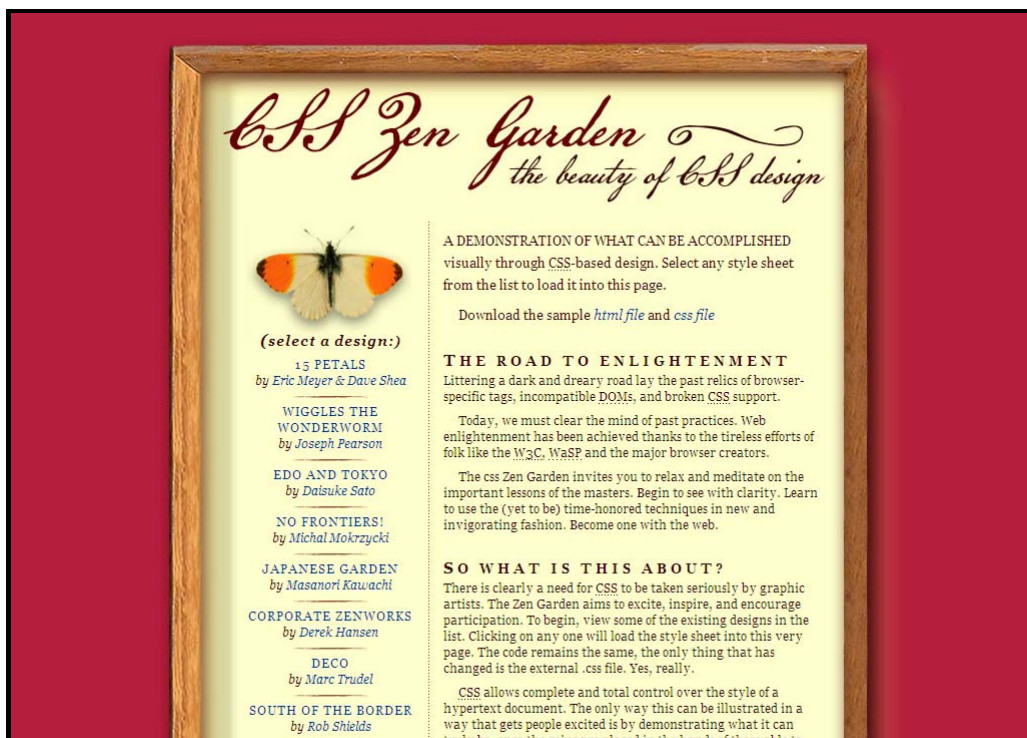
The Beauty of CSS Design

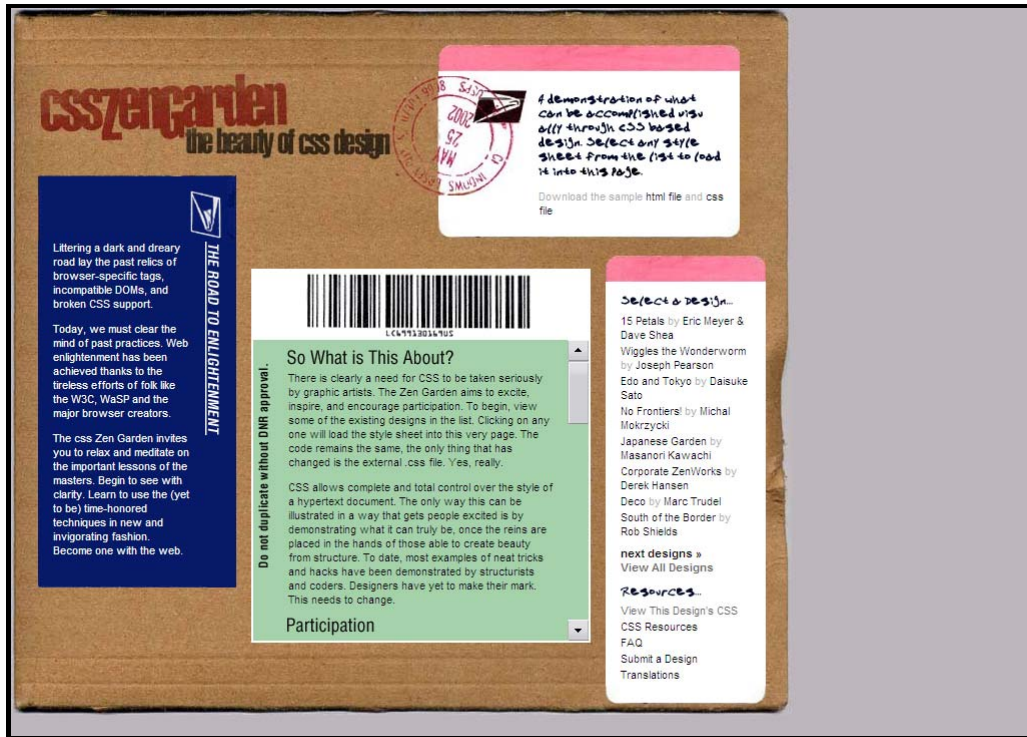
A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.

Download the sample [html file](#) and [css file](#)

It uses very simple HTML: div, h1, h2, h3, span, p, ul/li, a, acronym

As you look at each presented page, you have to keep repeating to yourself: this is the same HTML page.





zen garden

The road to enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WAFS, and the major browser creators.

The CSS Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.



so what is this about?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by demonstrating what it can truly be, once the reins are placed in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks have been demonstrated by structurists and coders. Designers have yet to make their mark. This needs to change.

15 Petals

by Eric Meyer & Dave Shea

Wiggles the Wonderworm

by Joseph Pearson

Edo and Tokyo

by Daisuke Sato

No Frontiers!

by Michal Mokrzycki

Japanese Garden

by Masanori Kawachi

Corporate ZenWorks

by Derek Hansen

Deco

by Marc Trudel

South of the Border

by Rob Shields

SELECT A DESIGN ARCHIVE RESOURCES

A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page. Download the sample [html file](#) and [css file](#).

dutchcelt



THE BEAUTY OF CSS DESIGN

A DEMONSTRATION OF WHAT CAN BE ACCOMPLISHED VISUALLY THROUGH CSS-BASED DESIGN. SELECT ANY STYLE SHEET FROM THE LIST TO LOAD IT INTO THIS PAGE.

DOWNLOAD THE SAMPLE HTML FILE AND CSS FILE

THE ROAD TO ENLIGHTENMENT

LITTERING A DARK AND DREARY ROAD LAY THE PAST RELICS OF BROWSER-SPECIFIC TAGS, INCOMPATIBLE DOMS, AND BROKEN CSS SUPPORT.

TODAY, WE MUST CLEAR THE MIND OF PAST PRACTICES. WEB ENLIGHTENMENT HAS BEEN ACHIEVED THANKS TO THE TIRELESS EFFORTS OF FOLK

15 PETALS

BY ERIC MEYER & DAVE SHEA

WIGGLES THE WONDERWORM

BY JOSEPH PEARSON

EDO AND TOKYO

BY DAISUKE SATO

NO FRONTIERS!

BY MICHAL MOKRZYCKI

JAPANESE GARDEN

BY MASANORI KAWACHI

CSS ZEN GARDEN PRESENTS

A DEMONSTRATION OF WHAT CAN BE ACCOMPLISHED VISUALLY THROUGH CSS-BASED DESIGN. SELECT ANY STYLE SHEET FROM THE LIST TO LOAD IT INTO THIS PAGE.

DOWNLOAD THE SAMPLE HTML FILE AND CSS FILE

STARRING
IN ORDER OF APPEARANCE

INVASION OF THE BODY SWITCHERS
by ANDY CLARKE

GOLDEN CUT
by PETR STANICEK

THE HALL
by MICHAEL SIMMONS

NEAT & TIDY
by OLI DALE

CUBE GARDEN
by MASANORI
KAWACHI

DJ STYLE
by RAMON BISPO

THE FINAL ENDING
by RAY HENRY

CONTEMPORARY NOUVEAU
by DAVID HELSING

INVASION OF THE BODY SWITCHERS



THE ROAD TO
ENLIGHTENMENT

THE CSS ZEN GARDEN INVITES YOU TO RELAX AND MEDITATE ON THE IMPORTANT LESSONS OF THE MASTERS. BEGIN TO SEE WITH CLARITY. LEARN TO USE THE (YET TO BE) TIME-HONORED TECHNIQUES IN NEW AND INVIGORATING FASHION. BECOME ONE WITH THE WEB.



4: Warning about old browsers

CSS implementations are now quite good, but older browsers had a variety of mistakes. Unfortunately, some browser manufacturers want to offer backwards compatibility with those buggy old browsers. So they have two modes: compliant mode, and legacy mode.

To decide which mode to use they look at the document.

5: Legacy and compliant modes

- For XML it is always in compliant mode
- For HTML it is compliant mode if you include a DOCTYPE at the top of the document, for instance:

```
<!DOCTYPE HTML
  PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<!DOCTYPE HTML
  PUBLIC "-//W3C//DTD HTML 4.0 Strict//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

If you want your CSS to work right, always include a DOCTYPE at the head of your document.

6: Warning about XHTML/XML

XHTML is an XML application: it is XML.

However, it was designed to be sendable to old-fashioned browsers as HTML; the old browsers think they have received HTML, the new ones think they have received XHTML; there are some differences, but in most cases you won't notice them.

So when we say above "For XML it is always in compliant mode", for XHTML we mean "when the browser thinks it has received XHTML and not HTML".

Note that certain HTML-only rules in CSS do *not* apply to XHTML.

Also remember that HTML is case-insensitive, and XHTML isn't, so a quick way to find out what your browser thinks it has got is the following CSS:

```
H1 {color: red}
h1 {color: green}
```

7: About Valid Documents

HTML was designed so that incorrect documents are corrected by the browser.

However, you can never know exactly how the browser has corrected your document. This means that you don't know the exact structure of your underlying document.

Which means that you can't be sure how your CSS will be applied.

Therefore: always ensure you have a valid document.

For instance, check it at the W3C validator at validator.w3.org

8: CSS2 Selectors

CSS2 added a number of new selectors to CSS, to allow finer selection of elements.

One thing to remember about CSS is that it has a document-order processing model. So several selectors are asymmetric: you can select the first child element, but not the last, you can select the following element, but not the previous, and so on.

9: CSS2 Selector overview

*

E > F

E:first-child

E + F

E[att]

E[att="value"]

E[att~="value"]

10: Universal selector

*: selects any element

This is useful for creating rules that are valid for all elements (for instance for a base style sheet that might work with several XML markup languages, where the root element is unknown).

```
* {color: black; background: white;
  font-family: sans-serif}
```

Note that a rule like

```
*.warning
```

is equivalent to

```
.warning
```

11: Child-of selector

E > F: selects any element F that is a child of E

For instance, these slides are in XHTML, and each slide is just a <div>.

I used to have for each slide

```
<div class="slide">
```

but now I can save typing, and have an unadorned <div>. Where I used to have the CSS rule:

```
div.slide {...}
```

I now have:

```
body>div {...}
```

which only selects divs which are a (direct) child of <body>

12: Alternative for child-of

Note that the rule

```
body div
```

also selects divs that are a child of <body>, but at any level, not just direct children. So if my slides themselves contain <div>s (which they do) then those divs would be selected too.

Another option I could have used was:

```
body div {... rules intended for slides... }  
body div div {... rules to undo the above rules ... }
```

but it is nicer not to have to 'undo' presentation caused by other rules if you can avoid it.

13: First-child selector

E:first-child: selects any E that is the first child of its parent

So for instance, the following only selects paragraphs that are the first child of their parent:

```
p:first-child
```

Note that this slide does not have such a paragraph. The first child is a heading, so the above rule wouldn't apply to any paragraph here.

Usually though you want to be more specific. For instance, to select the first slide of this set:

```
body > :first-child
```

which selects any element that is a direct child of <body> that is also a first-child of its parent. Since there is only one of those, the first child of the body, it selects that.

14: Adjacent-sibling selector

(Sibling = brother or sister)

E + F: selects any F that directly follows an E

For instance, this selects any paragraph that follows another paragraph (i.e., not the first paragraph):

```
p + p {...}
```

On this slide I have coloured all such paragraphs red.

This selects any paragraph that follows a paragraph that is the first-child, i.e. the second paragraph:

```
p:first-child + p {...}
```

15: Attribute selector

E[att]: any E that has attribute att

For instance, a special selector for paragraphs that have id attributes:

```
p[id] {color: green}
```

(More on this later when we talk about generated content)

16: Attribute value selector

E[att="value"]: where att has exactly the value

Since class selectors like this:

```
p.warning {color: red}
```

don't work for generic XML markup, this sort of selector can be used instead:

```
p[class="warning"] {color: red}
```

17: Attribute partial value selector

E[att~="value"]: where one of the words of att is the value

This is the one you should really use for simulating class selectors, because the class attribute may contain more than one class:

```
<p class="warning sidebar">...  
  
[class~="warning"] {color: red}  
[class~="sidebar"] {float: left; ... }
```

18: Microsoft Internet Explorer

Note that although most modern browsers support CSS 2, Microsoft Internet Explorer is not a 'modern browser' in this respect. In particular, almost none of the CSS2 selectors work in IE (:hover does).

This means that you can do special effect for other browsers, that IE won't see. For instance:

```
html>body {... IE won't see these rules...}
```

(Look at [CSS Zen Garden: Gemination](#) in IE and a modern browser for a surprising example of this)

19: Example of CSS1-explicit display

One use of this difference is to display a warning.

For instance these slides have a warning at the beginning that is only displayed with IE (and other CSS1-only browsers) pointing out to people that they won't see the slides as they are intended. CSS2 browsers don't display the warning.

```
<p class="csscheck">This document is written in XHTML and CSS2;  
while you will be able to read the content acceptably,  
you need a better browser than the one you are using  
to see this document properly.</p>  
  
.csscheck {color: red; font-size: 150%; width: 50%;  
margin-left: 1em; border: thick red solid}  
  
body>div>.csscheck {display: none}
```

Since the second rule is more specific, its rules override the less-specific set. (i.e., the warning still has a border and so on, but they are just not displayed)

20: Practical

21: Positioning

There are three main ways of positioning in CSS

- Float
- Absolute
- Relative

(Other lesser ways of positioning include using vertical-position and using margins)

24: Float left and right:

And here it is justified:

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.



At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.



25: Floating text

You can float anything, not just images. For instance blocks of text.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum.

Sidebar

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.

sanctus est Lorem ipsum dolor sit.

This is done by putting the sidebar in a `<div class="sidebar">`, and then using

```
.sidebar { float: left; width: 30%;  
            background-color: yellow;  
            margin: 0 1em 1em 0 }
```

26: Advantages of float

The nice thing about float is that the floated content stays within the main flow of the document, surrounding text just avoids it automatically.

The only thing you have to take care of is the occasional

```
clear: all
```

for following content that must clear the floated content.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum.

Sidebar

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.

27: Clearing

Beware of this sort of problem, where the floated object is longer than the text:

28: Relative positioning

Relative addressing takes the positioned content and moves it relative to where it would originally have been. It leaves a 'hole' where it would have been:

Relative positioning

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.

This was done with:

```
.relative {position: relative; top: 5em; left: 5em}
```

29: Absolute positioning

On the other hand, absolute positioning takes the content out of the flow, moves it, and 'closes up' the gap, as if it had never been there:

Absolute positioning

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit.

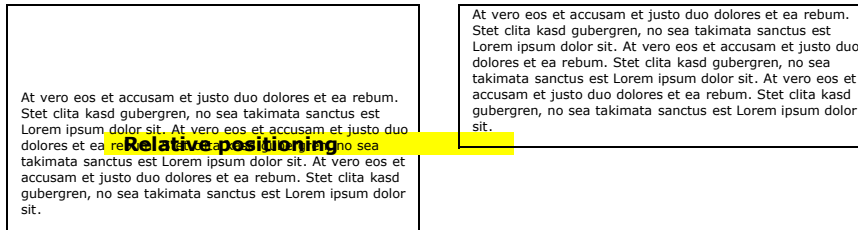
This was done with:

```
.absolute {position: absolute; top: 5em; left: 5em}
```

Note that the heading is moved in relation to an absolute point, not the point where it came from.

30: Note about apparent transparency

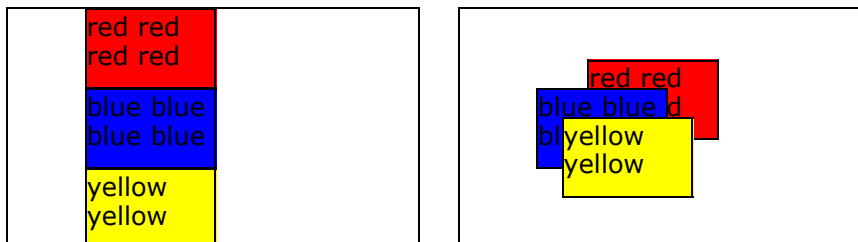
In the above examples it looks like relative positioning produced a transparent yellow heading and absolute positioning produced an opaque yellow heading:



However, this is just an effect of where in the document the text appears. The text after the relatively positioned heading is later in the document, and so is drawn on the screen later, i.e. over the earlier text. The absolutely positioned heading is also drawn over text that was drawn earlier. This therefore gives the appearance of transparency in one case and opaqueness in the other. Look at the borders above to see this effect more clearly.

31: Layering 1

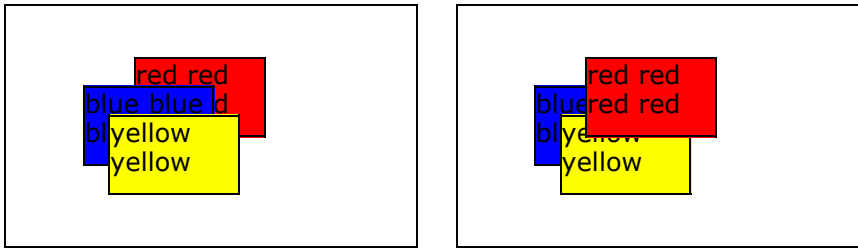
If you want to affect such layering explicitly, then you can use the `z-index` property. First let's relatively position some blocks.



```
.layer1 { background-color: red; position: relative;
           top: 2em; left: 2em; }
.layer2 { background-color: blue; position: relative;
           top: 0; left: 0; }
.layer3 { background-color: yellow; position: relative;
           top: -2em; left: 1em; }
```

32: Layering 2

Now we can apply the z-index property to the first block to make it be painted last:



```
.layer1, .layer1t { background-color: red; position: relative;
                    top: 2em; left: 2em; z-index: 1 }
.layer2 { background-color: blue; position: relative;
          top: 0; left: 0; z-index: 1 }
.layer3 { background-color: yellow; position: relative;
          top: -2em; left: 1em; z-index: 1 }
.layer1t { z-index: 2 }
```

33: Absolutely positioned to what?

Admission: I had to cheat slightly in the slides with an absolutely positioned heading. Absolutely positioned elements are positioned 'absolutely' relative to the closest positioned ancestor element, or otherwise relative to the root element.

So if I had done nothing, the absolutely positioned title in that slide would have appeared somewhere on the first slide. So I positioned all slides relatively, but by (0, 0) so that they didn't move, but so that absolutely positioned elements within them would move within that slide.

34: Summary of positioning

So to summarise:

- Relatively positioned elements are positioned relative to their original position.
- Absolutely positioned elements are positioned relative to their 'origin', which is the closest positioned ancestor element, or otherwise the root element.

Put another way: when an element is positioned, all its children move with it.

35: Overflow

Values: visible, hidden, scroll, auto, inherit
Default: visible

- Visible: show even if it doesn't fit
- Hidden: hide what doesn't fit
- Scroll: use scroll bars regardless
- Auto: use scrollbars if needed

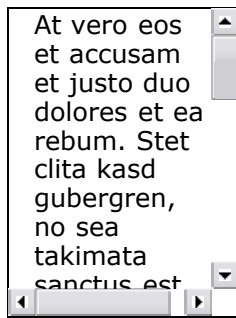
36: Overflow visible

At vero eos et
accusam et
justo duo
dolores et ea
rebum. Stet
clita kasd
gubergren, no
sea takimata
sanctus est
Lorem ipsum
dolor sit. At
vero eos et
accusam et
justo duo
dolores et ea
rebum. Stet
clita kasd
gubergren, no
sea takimata
sanctus est
Lorem ipsum
dolor sit. At

37: Overflow hidden

At vero eos et
accusam et
justo duo
dolores et ea
rebum. Stet
clita kasd
gubergren, no
sea takimata
sanctus est
Lorem ipsum
dolor sit. At

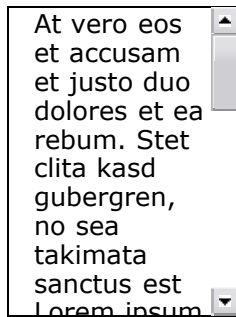
38: Overflow scroll



At vero eos
et accusam
et justo duo
dolores et ea
rebum. Stet
clita kasd
gubergren,
no sea
takimata
sanctus est

This screenshot shows a text area with a vertical scrollbar on the right and a horizontal scrollbar at the bottom. The text is truncated at the bottom, with the last line being "sanctus est".

39: Overflow auto



At vero eos
et accusam
et justo duo
dolores et ea
rebum. Stet
clita kasd
gubergren,
no sea
takimata
sanctus est
Lorem ipsum

This screenshot shows a text area with a vertical scrollbar on the right and a horizontal scrollbar at the bottom. The text is truncated at the bottom, with the last line being "Lorem ipsum".

40: Hovering

The basis of hovering is for changing colours or similar. Like this:

```
a:hover {background-color: yellow}
```

But it offers more possibilities, such as this: [Hover here]

41: Hovering

The basis is to have the pop-up text in the element that you are going to hover over:

```
<span class="popup">[Hover here]<span>Bla bla</span>  
</span>
```

Then the default case is to make the pop-up text display: none:

```
.popup span {display: none}
```

42: Hovering

```
<span class="popup">[Hover here]<span>Bla bla</span>
</span>
```

Then, when we hover, we make the span visible, and position it as necessary:

```
.popup:hover span {
  display: block; width: 10em;
  background-color: yellow;
  border: dotted black thin;
}
```

[Hover here]

You see that later content is pushed away, because the popup text is being displayed where it is in the content.

43: Hovering with relative position

```
<span class="popup">[Hover here]<span>Bla bla</span>
</span>
```

```
.popup:hover span {
  display: block; width: 10em;
  background-color: yellow;
  border: dotted black thin;
  position: relative; left: 30%; right: -10em;
}
```

[Hover here rel]

You see that later content is pushed away, because the popup text is being displayed where it is in the content, and then moved relative.

44: Hovering with absolute positioning

```
<span class="popup">[Hover here]<span>Bla bla</span>
</span>
```

```
.popup:hover span {
  display: block; width: 10em;
  background-color: yellow;
  border: dotted black thin;
  position: absolute; left: 30%; top: 12em;
}
```

[Hover here abs]

You see that later content is not pushed away, because the popup text is being taken out of the flow.

45: Practical

46: Numbering

CSS allows you to automatically number elements. For instance, in an alternate stylesheet, these slides are automatically numbered. You can use numbering for such things as ordered lists, for numbering headings, or for numbering diagrams or equations.

id: generated-content **Generated content**

The first part of the solution is to do with *generated content*:

```
p:before {content: "<<< "}  
p:after  {content: " >>>"}
```

```
<<< One >>>  
<<< Two >>>  
<<< Three >>>
```

Here is a nice trick:

```
[id]:before { font-size: small;  
              content: "id: " attr(id) " " }
```

48: Counters

The next part of the solution is counters.

```
ol { counter-reset: item }  
ol>li {counter-increment: item }  
  
ol>li:before {counter(item) ". " }
```

Counter-reset creates a counter if necessary, and sets it to 0.

49: Nested counters

Nesting creates a new level of counter. The increment applies then to the most recent one:

```
<ol>  
<li> ..... (1)  
<li> ..... (2)  
  <ol> ... (2,0)  
    <li> ... (2,1)  
    <li> ... (2,2)  
  </ol>  
</li> ..... (3)
```

50: Nested counters

You can generate a list of all the counters with the same name:

```
li:before {content: counters(item, ".") ": " }
```

Which will generate something like:

```
2.2:
```

51: Resetting to different values

You can also do things like

```
counter-reset: item 20
```

and

```
counter-reset: chapter 0 section 0
```

etc.

52: Formatting the counter

You can format the counter in different ways:

```
content: counter(item, lower-latin) ". "
```

will give i. ii. iii. iv. v. etc.

The second parameter is just the same as available for 'list-style-type':

disc | circle | square | decimal | decimal-leading-zero | lower-roman |
upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian
| lower-alpha | upper-alpha | none | inherit

This also works for the 'counters' function:

```
content: counters(item, ".", upper-latin)
```

Which will generate things like:

II.II II.III II.IV II.V

53: Counter example

```
h2 {counter-increment: chapter;  
    counter-reset: section figure eqn}  
h2:before {content: "Chapter " counter(chapter) ": " }  
.eqn {counter-increment: eqn}  
.eqn:after {  
    content: " ..." counter(chapter) "." counter(eqn)  
}
```

$E = MC^2$ eqn 53.1

Appendixes as well

```
h2.appendix {counter-increment: appx}  
h2.appendix:before {  
    content: "Appendix " counter(appx, upper-alpha) ": "  
}
```

54: Practical

55: Images

Essentially you have the choice between three different types of image:

- gif
- jpg
- png

Each type has its own advantages and disadvantages, and so the choice of which to use should depend on your needs.

An essential property of each format is that they are compressed: the storage and bandwidth needed is less than for the full image.

But beware: the image gets unpacked once it is used, so it can use much more memory than the compressed size. (One website I visited brought my machine to a crawl, and it turned out that it had a tiny (in bytes) image as background image. But when unpacked it was very large, and used up all my swap space.)

56: GIF

Properties:

- Uses a colour map: available colors are from 2 to 256 in powers of 2.
- So each pixel uses anything from 1 bit to 8 bits.
- One colour may be designated as transparent.
- Uses Lempel-Ziff-Welch compression which identifies patterns in the byte stream to compress the result.
- Used to be patent-encumbered, but the patent has expired.

This means:

- Low number of colours available
- Otherwise exact reproduction of image
- So good for text
- Minimal transparency control

57: JPG

Properties:

- Uses full colour, 24 bits per pixel, 8 bits per colour, so millions of colours available.
- Uses 'psychological' (lossy) compression, based on properties of how we see.
- If you enlarge a jpg image, 'artefacts' become visible.
- The more you compress, the more visible the artefacts.
- No transparency.

This means:

- Large number of colours
- Non-exact reproduction of colours
- Good for photos
- Bad for text
- No transparency

58: PNG

Properties of PNG:

- Several forms, but 2 basic: colour map (like GIF), or colour (like JPG)
- Lossless compression, usually better than GIF
- 32 bits per pixel: 8 per colour, plus 8 for transparency (double this also possible)

This means:

- Good for text
- Can be smaller than GIF, but can also be large when using full colour
- Excellent transparency control

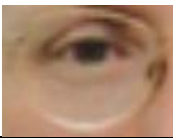
59: Comparison



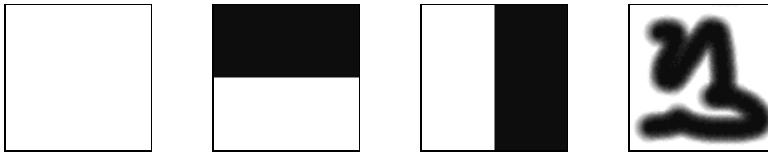
60: Comparison of JPG compressions



Low and high compression; 5 kB vs 32 kB



61: Comparison of GIF sizes



Four gifs: size 941 1000 1102 2367 bytes

These differences are due to the different amount of repetition in the file.

62: Comparison of colour depth in GIF



63: Transparency trick

An oft-used trick is to make a sort of chess-board style of the pixels in the image, where alternate pixels are transparent. These are all the same image, on different coloured backgrounds:



64: Transparency trick



65: Image replacement techniques

A major technique that is used in CSS Zen Garden is of replacing some text with an image.

This has an advantage over traditional uses of images for text, that the text is still in the document so that Google can see it, and so can blind people.

Let's look at an example for replacing a heading with an image:

66: *images for Headlines*

In the HTML, use a heading, with the text in a span:

```
<h1><span>Images for headlines</span></h1>
```

In the CSS, turn off the span:

```
h1 span {display: none}
```

and add a background image to the h1:

```
h1 { background-image: url(text.gif);  
      background-repeat: no-repeat;  
      background-position: 50% 50%;  
      height: 80px; width: 80px; }
```

67: Image replacement problems

There can be some accessibility problems with this technique.

Some screen readers don't see text with 'display: none'.

No one sees anything if images have been turned off.

An alternative technique is to position the 'hidden' text off the screen (note that the span is not necessary):

```
h1 { text-indent: -100cm; ... }
```

Another technique is to move something over the top of the text:

```
<h1>Images for headlines<span></span></h1>  
  
h1 {position: relative}  
h1 span { position: absolute; top: 0; left: 0; display: block}
```

and then use the background image.

68: Round corners

You can use background images to simulate round corners.

At vero eos et accusam et justo
duo dolores et ea rebum. Stet clita
kasd gubergren, no sea takimata
sanctus est Lorem ipsum dolor sit.
At vero eos et accusam et justo
duo dolores et ea rebum.

```
.curved { background: transparent url(curve.GIF)
           no-repeat 50% 0%;
           width: 211px; padding-top: 8px;
           font-size: 80%; margin-left: 10% }
}
```

69: Shadows

You can use background images to do shadow effects.

Image: ■■

At vero eos et accusam et justo
duo dolores et ea rebum. Stet clita
kasd gubergren, no sea takimata
sanctus est Lorem ipsum dolor sit.
At vero eos et accusam et justo
duo dolores et ea rebum.

```
.shadow { background: transparent url(shadow.png) repeat-y 100% 100%;
           width: 211px; padding-top: 8px; padding-right: 2em;
           font-size: 80%; margin-left: 10% }
}
```

70: Practical

71: Fluid Design

One of the big problems with many current web sites is the fixed design.

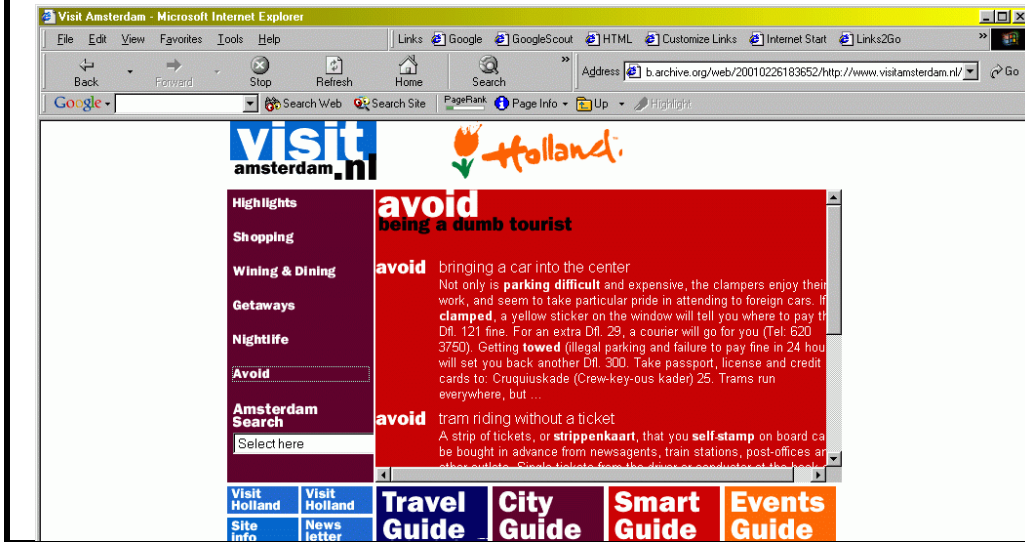
Part of the problem is that designers have grown up designing for paper, where you have complete control.

On the web, whatever designers may want, you no longer have complete control: you don't know the size of the screen, let alone the size of the window, or even the fonts available, let alone their sizes.

To best serve the interests of the user, you should attempt to utilise the screen real-estate as well as possible.

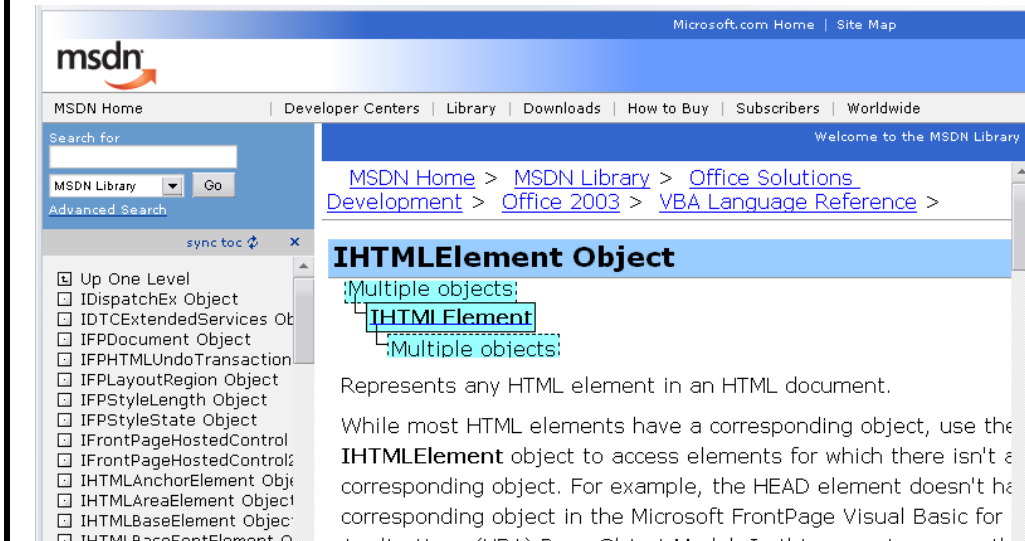
72: How not to do it

Here is an example site, luckily now changed, of a site underutilising the space available:



73: How not to do it (2)

Here is a site that overuses the space, making it very hard to read. In particular you have to scroll horizontally to read the text.



74: How to do it

The aim is to use the space as well as possible.

- Fixed text areas (for instance navigation lists) may stay a fixed width
- But flowing text areas should be allowed to expand and contract, within a range.
- Maybe use float, that will then be floated away when the available space becomes too small

75: Reminder about sizes

Don't use pixels for sizes, except for images with an inherent size, because you don't know the resolution of the screen that anyone is using.

For instance, `font-size: 12px` will give you

- 12 point on a 72dpi screen
- 9 point on a 96dpi screen
- 6.9 point on a 125dpi screen

76: Sizes

Similarly, avoid using point sizes as well:

- Many machines are not properly set up (the operating system doesn't know the screen resolution)
- Many people can't see text below a certain size, and so increase the base font-size

Respect these people: try to use relative font sizes. Use font-size: 100% for paragraph text, font-size: 120% or so for headings, font-size: 80% for copyright statements, etc.

77: Sizes for width

Similarly, don't use pixels for widths either, for the same reasons

Try whenever possible to express widths as percentages (which refers to the parent element) or as ems (which then refers to the current content).

78: Avoid too narrow or too wide text regions

Good typesetting practise requires you to match your line spacing to the length of the text lines. Long text lines should have wide line spacing, otherwise the reader is likely to get lost when going to the start of the next line.

However, CSS doesn't offer the facility to increase line spacing automatically with line length, so you should therefore restrict the length of your text lines.

But beware: too short lines are unreadable as well.

79: Example

Solemnly he came forward and mounted the round gunrest. He faced about and blessed gravely thrice the tower, the surrounding country and the awaking mountains. Then, catching sight of Stephen Dedalus, he bent towards him and made rapid crosses in the air, gurgling in his throat and shaking his head. Stephen Dedalus, displeased and sleepy, leaned his arms on the top of the staircase and looked coldly at the shaking gurgling face that blessed him, equine in its length, and at the light untoursured hair, grained and hued like pale oak.

Solemnly he came forward and mounted the round gunrest. He faced about and blessed gravely thrice the tower, the surrounding country and the awaking mountains. Then, catching sight of Stephen Dedalus, he bent towards him and made rapid crosses in the air, gurgling in his

Solemnly he came forward and mounted the round gunrest. He faced about and blessed gravely thrice the tower, the surrounding country and the awaking mountains. Then, catching sight of Stephen Dedalus, he bent towards him and made rapid crosses in the air, gurgling in his throat and shaking his head. Stephen Dedalus, displeased and sleepy, leaned his arms on the top of the staircase and looked coldly at the shaking gurgling face that blessed him, equine in its length, and at the light untoursured hair, grained and hued like pale oak.

80: min-width and max-width

These two properties allow you to constrain the width within certain bounds.
Example:

Widths: this example paragraph has a width of 70%, a max-width of 30em and a min-width of 20em.

|
|

81: Some examples

Example 1

|
|

82: Some examples

Example 2

|
|

83: Some examples

Example 3

|
|

84: Some examples

Example 4

|
|

85: Some examples

Example 5

|
|

86: Device-independent design



There are now more browsers on mobile phones than on PCs.

Web sites seldom cater for small screen browsers, and yet it is so easy.

87: Device-independent design

Use device dependent stylesheets:

```
<link href="basic.css" rel="stylesheet" media="all" />
<link href="screen.css" rel="stylesheet" media="screen" />
<link href="phone.css" rel="stylesheet" media="handheld" />
```

or

```
<style type="text/css" media="phone">
...
</style>
```

or

```
<style type="text/css" media="all">
... general rules ...
@media screen { ... specific rules ... }
@media handheld { ... specific rules ... }
</style>
```

Use fluid design.

88: **By the way**

It is quite all right to use point sizes in a print style sheet

89: **Practical**

90: Overview of properties, with examples and defaults

- font-*:
 - family (**Futura**, ..., serif, sans-serif, cursive, fantasy, monospace)
 - style (**normal**, italic, oblique)
 - variant (**normal**, small-caps)
 - weight (**normal**, bold, bolder, lighter, 100, ..., **400**, ..., 900)
 - size (10pt, 120%, small, **medium**, large, smaller, larger, ...)
- color (**red**, ..., #f00, #ff0000, rgb(255,0,0), rgb(100%, 0, 0), ...)
- background-*:
 - color (**transparent**, red, black, white, gray, silver, red, maroon, yellow, olive, green, lime, blue, navy, purple, aqua, fuchsia, teal, ...)
 - image (none, url(back.gif))
 - repeat (**repeat**, no-repeat, repeat-x, repeat-y)
 - attachment (**scroll**, fixed)
 - position (**0% 0%**, top left, center, center left, bottom right, ...)
- line-height (**normal**, 120%, ...)
- word-spacing, letter-spacing (normal, 1%, 1px, ...)
- vertical-align (**baseline**, sub, super, 10%, top, text-top, middle, ...)
- text-*:
 - decoration (**none**, underline, overline, line-through, blink)
 - transform (**none**, uppercase, lowercase, capitalise)
 - align (justify, left, right, center)
 - indent (**0**, 4em, ...)
- display (**block**, inline, list-item, none)
- white-space (**normal**, pre, nowrap)
- list-style-*:
 - type (**disc**, circle, square, decimal, none, lower-roman, lower-alpha, ...)
 - image (url(sphere.gif), **none**)
 - position (inside, **outside**)
 - list-style (*type position <url>*)

91: Overview of box properties

- margin-*: top, right, bottom, left (**0**, auto, 2em, 3pt, 1%, ...)
- padding-*: top, right, bottom, left (**0**, 2em, 3pt, 1%, ...)
- border-*:
 - width⁴ (thin, **medium**, thick, 2pt, ...)
 - style (**none**, dotted, dashed, solid, double, ...)
 - color⁴ (...)
 - top, right, bottom, left (*width style colour*)
 - top-width, bottom-width, right-width, left-width (**medium**, ...)
- margin⁴ (*top right bottom left*)
- padding⁴ (*top right bottom left*)
- border (*width style color*)
- height, width (**auto**, 100px, 15em, 50%, ...)
- float (**none**, left, right)
- clear (**none**, left, right, both)

92: Web Resources for CSS, XML and XHTML

- A list of known books about CSS, online resources, supporting browsers, and editors: <http://www.w3.org/Style/CSS>.
- The CSS1 Recommendation: <http://www.w3.org/TR/REC-CSS1>
- The CSS2 Recommendation: <http://www.w3.org/TR/REC-CSS2/>
- A CSS1 Quick reference: <http://www.cwi.nl/~steven/www/css1-qr.html>
- XHTML: <http://www.w3.org/TR/xhtml1/> with more information at <http://www.w3.org/Markup>
- XML: <http://www.w3.org/TR/REC-xml> with more information at <http://www.w3.org/XML/>
- An excellent book on CSS2, is by two of its creators, Håkon Lie and Bert Bos. It is *Cascading Style Sheets: Designing for the Web*, published by Addison-Wesley. The latest edition was written entirely in XHTML and CSS.
- Another, about CSS Zen Garden, is *The Zen of CSS Design*, by Dave Shea and Molly E. Holzschlag, published by New Riders.
- Validate your CSS: <http://jigsaw.w3.org/css-validator/>
- Validate your HTML and XHTML: <http://validator.w3.org/>
- Tidy up your HTML, making it more amenable for CSS, and convert it to XHTML 1.0: <http://www.w3.org/People/Raggett/tidy/>
- Test your browser for CSS compliance: <http://www.w3.org/Style/CSS/Test/>
- A set of style sheets for HTML designed by a graphic-designer: <http://www.w3.org/StyleSheets/Core/>
- How well (or badly) browsers implement CSS, feature by feature: <http://www.webreview.com/style/css1/charts/mastergrid.shtml>
- Workarounds for browser bugs: <http://css.nu/pointers/bugs.html>
- Google's directory of resources: <http://directory.google.com/Top/Computers/Programming/Internet/CSS/>.
- The 'House of Style': http://www.westciv.com/style_master/house/
- A webzine of standards-based web-building techniques: <http://www.alistapart.com/stories/>.
- Standards evangelism: <http://www.webstandards.org/>.
- Some examples of CSS-based sites: W3C, Wired, webstandards.org, A List Apart, a Lycos redesign at <http://jscript.dk/lycos/2/>, XHTML2 at <http://w3future.com/weblog/gems/xhtml2.xml>.

93: Quick Reference to Cascading Style Sheets, level 1

This version is based on:

<http://www.w3.org/TR/REC-CSS1>

Latest version:

<http://www.w3.org/TR/WD-css1.html>

Author:

Steven Pemberton (Steven.Pemberton@cwi.nl)

94: Syntax

a b c

a is followed by b is followed by c, in that order.

a | b

either a or b must occur

a || b

either a or b or both must occur, in any order

[a b]

brackets, used for grouping

a?

a is optional

a*

a is repeated 0 or more times

a+

a is repeated 1 or more times

a{1,4}

a is repeated at least once and at most 4 times.

Juxtaposition is stronger than the double bar, and the double bar is stronger than the bar. Thus "a b | c || d e" is equivalent to "[a b] | [c || [d e]]".

95: Definitions

Block-level elements

an element which has a line break before and after (e.g. <H1>, <P>)

Replaced element

An element which is replaced by content pointed to from the element.
E.g., .

96: Properties

In each definition

- the default value is shown in bold, or given separately
- values apply to all elements unless otherwise stated
- properties are inherited unless the property name is marked with a star "*".

97: 5.2 Font properties

5.2.2 font-family

[[<family-name> | <generic-family>],]* [<family-name> | <generic-family>]

Initial: UA specific

<generic-family>

serif | sans-serif | cursive | fantasy | monospace

5.2.3 font-style

normal | italic | oblique

5.2.4 font-variant

normal | small-caps

5.2.5 font-weight

normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

5.2.6 font-size

<absolute-size> | <relative-size> | <length> | <percentage>

Initial: medium

Percentage values: relative to parent element's font size

<absolute-size>

xx-small | x-small | small | medium | large | x-large | xx-large

<relative-size>

larger | smaller

5.2.7 font

[<font-style> || <font-variant> || <font-weight>]? <font-size> [/ <line-height>]? <font-family>

Initial: not defined for shorthand properties

Percentage values: allowed on <font-size> and <line-height>

98: 5.3 Color and background

5.3.1 color

<color>

Initial: UA specific

5.3.2 background-color*

<color> | transparent

5.3.3 background-image*

<url> | none

5.3.4 background-repeat*

repeat | repeat-x | repeat-y | no-repeat

5.3.5 background-attachment*

scroll | fixed

5.3.6 background-position*

[<percentage> | <length>]{1,2} | [top | center | bottom] || [left | center | right]

Applies to: block-level and replaced elements

Initial: 0% 0%

Percentage values: refer to the size of the element itself

5.3.7 background*

<background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>

Initial: not defined for shorthand properties

Percentage values: allowed on <background-position>

99: 5.4 Text properties

5.4.1 word-spacing

normal | <length>

5.4.2 letter-spacing

normal | <length>

5.4.3 text-decoration*

none | [underline || overline || line-through || blink]

Inherited: no, but see clarification below

5.4.4 vertical-align*

baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage>

Applies to: inline elements

Percentage values: refer to the 'line-height' of the element itself

5.4.5 text-transform

capitalize | uppercase | lowercase | **none**

5.4.6 text-align

left | right | center | justify

Applies to: block-level elements

Initial: UA specific

5.4.7 text-indent

<length> | <percentage>

Applies to: block-level elements

Initial: 0

Percentage values: refer to parent element's width

5.4.8 line-height

normal | <number> | <length> | <percentage>

Percentage values: relative to the font size of the element itself

100: 5.5 Box properties

5.5.1 margin-top*

<length> | <percentage> | auto

Initial: 0

Percentage values: refer to parent element's width

5.5.2 margin-right*

<length> | <percentage> | auto

Initial: 0

Percentage values: refer to parent element's width

5.5.3 margin-bottom*

<length> | <percentage> | auto

Initial: 0

Percentage values: refer to parent element's width

5.5.4 margin-left*

<length> | <percentage> | auto

Initial: 0

Percentage values: refer to parent element's width

5.5.5 margin*

[<length> | <percentage> | auto]{1,4}

Initial: not defined for shorthand properties

Percentage values: refer to parent element's width

5.5.6 padding-top*

<length> | <percentage>

Initial: 0

Percentage values: refer to parent element's width

5.5.7 padding-right*

<length> | <percentage>

Initial: 0

Percentage values: refer to parent element's width

5.5.8 padding-bottom*

<length> | <percentage>

Initial: 0

Percentage values: refer to parent element's width

5.5.9 padding-left*

<length> | <percentage>

Initial: 0

Percentage values: refer to parent element's width

5.5.10 padding*

[<length> | <percentage>]{1,4}

Initial: 0

Percentage values: refer to parent element's width

5.5.11 border-top-width*

thin | **medium** | thick | <length>

5.5.12 border-right-width*

thin | **medium** | thick | <length>

5.5.13 border-bottom-width*

thin | **medium** | thick | <length>

5.5.14 border-left-width*

thin | **medium** | thick | <length>

5.5.15 border-width*

[thin | medium | thick | <length>]{1,4}

Initial: not defined for shorthand properties

5.5.16 border-color*

<color>{1,4}

Initial: the value of the 'color' property

5.5.17 border-style*

none | dotted | dashed | solid | double | groove | ridge | inset | outset

5.5.18 border-top*

<border-top-width> || <border-style> || <color>

Initial: not defined for shorthand properties

5.5.19 border-right*

<border-right-width> || <border-style> || <color>

Initial: not defined for shorthand properties

5.5.20 border-bottom*

<border-bottom-width> || <border-style> || <color>

Initial: not defined for shorthand properties

5.5.21 border-left*

<border-left-width> || <border-style> || <color>

Initial: not defined for shorthand properties

5.5.22 border*

<border-width> || <border-style> || <color>

Initial: not defined for shorthand properties

5.5.23 width*

<length> | <percentage> | **auto**

Applies to: block-level and replaced elements

Percentage values: refer to parent element's width

5.5.24 height*

<length> | **auto**

Applies to: block-level and replaced elements

5.5.25 float*

left | right | **none**

5.5.26 clear*

none | left | right | both

101: 5.6 Classification

5.6.1 display*

block | inline | list-item | none

5.6.2 white-space

normal | pre | nowrap

Applies to: block-level elements

5.6.3 list-style-type

disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none

Applies to: elements with 'display' value 'list-item'

5.6.4 list-style-image

<url> | **none**

Applies to: elements with 'display' value 'list-item'

5.6.5 list-style-position

inside | **outside**

Applies to: elements with 'display' value 'list-item'

5.6.6 list-style

<list-style-type> || <list-style-position> || <url>

Applies to: elements with 'display' value 'list-item'

Initial: not defined for shorthand properties

102: 6.1 Length units

<length>

[+|-]?<number><unit>

<number>

<digit>+[.<digit>]*?

<unit>

<absolute-unit> | <relative-unit>

<absolute-unit>

mm | cm | in | pt | pc

<relative-unit>

em | ex | px

103: 6.2 Percentage units

<percentage>

<number>%

104: 6.3 Color units

<color>

<color-name> | <rgb>

<color-name>

aqua | black | blue | fuchsia | gray | green | lime | maroon | navy |
olive | purple | red | silver | teal | white | yellow

<rgb>

#<hex><hex><hex> |

#<hex><hex><hex><hex><hex><hex> |

rgb(<number>, <number>, <number>) |

rgb(<percentage> <percentage>, <percentage>)

6.4 URL

<url>

url(<text>)