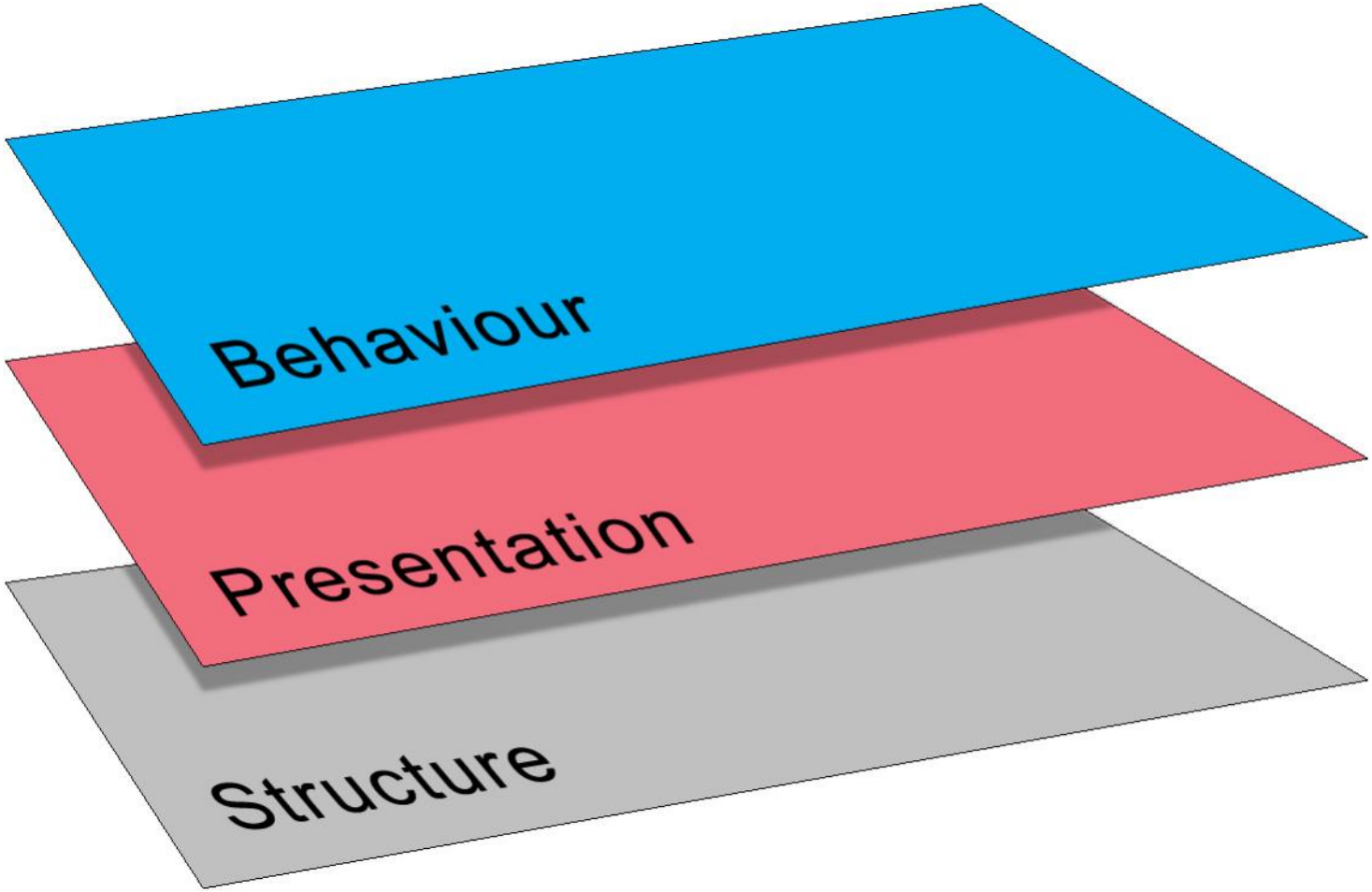
The Presentation Layer (Cascading Style Sheets)

Webpage Design

Anatomy of a webpage



What is CSS?

- Separating content from presentation
- Smaller files = quicker load
- Much greater control over formatting
- Easier site maintenance
- Improves accessibility
- The Web is in transition between CSS 2.1 and CSS 3
- This is more of an evolutionary process than the transition from XHTML to HTML5

http://friendlybit.com/css/beginners-guide-to-css-and-standards

How does it work?

Markup (XHTML or HTML5) index.html

+

Style Rules (CSS) base.css

=

Rendered result in browser

Style Rules

Selector **Declaration**

```
h1 {color: red}
```

Property **Value**

A CSS style rule is composed of a *Selector* and one or more *Declarations*. The example above has just one declaration. Each declaration has a *Property* and a *Value*. The declaration is always enclosed in curly braces and the colon character is used as a separator between the property and its value. This rule says that all h1 elements will have the colour red (note the spelling of this property).

http://www.w3schools.com/css/css_syntax.asp

http://css.maxdesign.com.au/selectutorial/rule.htm

Multiple Declarations

```
h1 {
   color: red;
   background-color: yellow;
   margin-bottom: 25px;
}
```

One CSS rule can contain many declarations.

The **semicolon** character is used as a separator between property/value pairs (declarations) within a single rule. The rule above has 3 declarations and requires 2 semicolons. Most coders add a semicolon after the last declaration even though it's not needed.

http://css.maxdesign.com.au/selectutorial/rules_group_dec.htm

Grouping Selectors

```
h2, h3, h4 {
    font-family: arial, helvetica, sans-serif;
    font-weight: normal;
    color: #343F4E;
}
```

In the above example, the same rule (consisting of 3 declarations) applies to h2, h3 and h4. The **comma** is used as a separator between selectors.

Note that the colour is defined using hexadecimal and not a colour name as in the previous examples.

http://css.maxdesign.com.au/selectutorial/rules_group_sel.htm

Pseudo-Class Rules

```
a {
   color: #009;
   text-decoration: none;
}
a:hover {
   text-decoration: underline;
}
```

Some HTML elements have different states or child elements.

The Anchor element has a number of different states and these can be styled independently using a CSS pseudo-class.

In this example, the hover state is styled differently from the normal link state.

http://www.w3schools.com/css/css_pseudo_classes.asp

http://css.maxdesign.com.au/selectutorial/selectors_pseudo_class.htm

Element Selector Rule in Action

CSS Rule

```
h2 {color: red;}
```

Markup

```
<h2>Heading</h2>
```

Rendered Result

Heading

All text within <h2> tags will be rendered red.

Types of CSS Selector

Element Type Selector `h1 {color: red;}`

Class Selector `.review {color: red;}`

ID Selector `#footer {color: red;}`

CSS allows for a number of selector types in addition to the element type selector. The most common are shown above. Class and ID names are just labels that allow CSS (and other technologies) to identify specific elements in a document. Class and ID names can be anything you like (provided they begin with a letter) but they should be *sensible* – not necessarily semantic.

CSS Selectors

Applying Classes and IDs

Classes and IDs may be applied to any HTML element. In both cases, they are applied as attributes, using the class or ID name as the attribute value.

```
<p class="review">Lorem ipsum...</p>
```

```
<p id="footer">Lorem ipsum...</p>
```

HTML elements may have more than one class. A space is used as a separator between each class name.

```
<p class="review book">Lorem ipsum...</p>
```

HTML elements may also have an ID *and* one or more classes...

```
<p id ="review" class="book">Lorem ipsum...</p>
```

…but never two IDs.

[The Difference Between ID and Class]

Class Selector Rule in Action

General Class Rule or... `.review {color: red;}`

Specific Class Rule `p.review {color: red;}`

Markup `<p class="review">Lorem ipsum...</p>`

Rendered Result

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam in nisi ac felis condimentum laoreet ac a eros. Vivamus commodo luctus massa sed consectetur. Aliquam elementum malesuada viverra.

In the examples above, the general class rule will apply to all elements with the "review" class but the specific class rule will apply only to paragraphs with the "review" class. In both cases, all text within <p> tags with the "review" class will be rendered red. There may be one or more paragraph elements with a "review" class in the document.

ID Selector Rule in Action

ID Selector Rule

```
#footer {color: red;}
```

Markup

```
<p id="footer">Copyright 2014</p>
```

Rendered Result

Copyright 2014

All text within a <p> tag with the "footer" ID will be rendered red. An ID is an unique identifier for a specific element – only one footer should exist in this document.

Recently, it has been suggested that the ID selector should be avoided and only class selectors should be used to target elements. This is because of a complication in CSS known as *specificity*. Essentially, ID selectors have precedence over class selectors and this can cause confusion in complex stylesheets where style rule conflicts occur.

CSS Specificity: Things You Should Know

What is a sensible name for a class?

For many years, it was considered good practice to use class and ID names that added meaning to our markup and this is entirely in keeping with our philosophy of creating semantically rich markup. So, a class name like:

```
<p class="book-review">This is a great book…</p>
```

Leaves us in no doubt that the paragraph is part of a book review as well as being a convenient selector for our CSS rule.

However, it has recently been pointed out (by Harry Roberts and others) that non-human agents (bots, screen-readers etc.) derive no additional meaning from class or ID names and they are therefore neither semantic nor non-semantic. So, giving descriptive names to classes and IDs is *sensible* but it is not *semantic*. How sensible is this class name?

```
<strong class="red">£9.99</strong>
```

What is a sensible name for a class?

```
<strong class="red">£9.99</strong>
```

There is nothing intrinsically wrong with this class name, it will work perfectly well but it is not a sensible name to use, even though it describes the effect of the rule it defines:

```
.red {color:red;}
```

The problem is that design is an organic process and things change. Let's say you decide that special offer prices should be displayed in blue and not red. We then have a style rule that looks like this:

```
.red {color:blue;}
```

This can only lead to confusion and possibly madness, so a much more sensible class name would be:

```
.special-offer {color:red;}
```

Read Harry Robert's chapter in Smashing Book #4 for more information.

Adding Styles to Webpages

Inline style

```
<h1 style="color:red">...</h1>
```

Embedded style in <head>

```
<style type="text/css">
h1{color: red;}
</style>
```

Linked style in <head>

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

Inline styles are added, using the *style* attribute, to html elements in the <body> of the document (not recommended). Embedded and Linked styles are added to the <head> of the document using the *style* and *link* elements respectively. Linked stylesheets are recommended in most cases.

http://matthewjamestaylor.com/blog/adding-css-to-html-with-link-embed-inline-and-import

Linked Stylesheets are Best

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

presentation structure structure + presentation

Linked or *external* stylesheets are considered best practice because they allow the separation of style from content (keeping the structure and presentation layers apart) and storing them in separate files. You may sometimes see @import used to import stylesheets rather than linking to them. The key reason you would do this is to hide styles from older browsers that don't understand @import. See the link below for more details.

http://webdesign.about.com/od/beginningcss/f/css_import_link.htm

A Linked Stylesheet

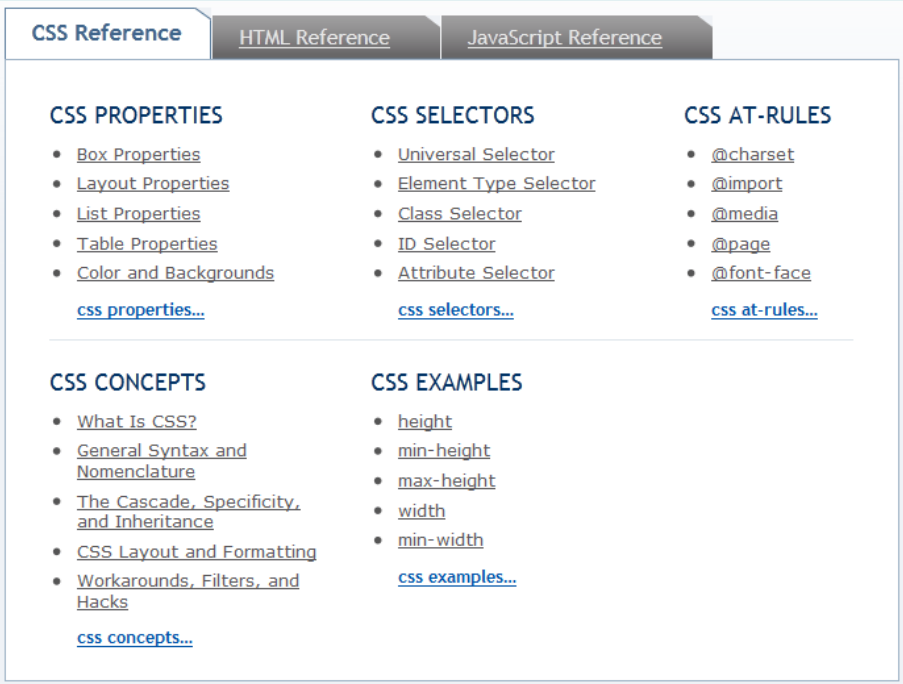
```
/* Page Layout */
body {
   padding-bottom:20px;
   border-top-width:10px;
   border-top-style:solid;
}
#wrapper {
   width:800px;
   margin:0 auto;
   padding-top:25px;
}
#branding {
   height:144px;
   margin-bottom:35px;
}
#content-main {
   width:552px;
   float:left;
   padding-right:25px;
   border-right-width:1px;
   border-right-style:solid;
   margin-bottom:20px;
}
/* Typography */
h1 {
   font-height:35px;
}
…
```

The column on the left is an extract from a linked stylesheet file. It's good practice to add comments to your stylesheet to help identify certain sections. Also, structure your CSS file in a logical way so that you can easily find rules.

Notice that this particular stylesheet contains no colour rules. That's because this stylesheet describes the layout of the page and a second stylesheet is used to define the colours. This is just the coder's personal preference.

More than one stylesheet can be linked to a webpage using additional <link> elements in <head>.

Which properties can I use?



Use a reference. The excellent online reference at Sitepoint is a good option. You can even use a book if you prefer that medium.

CSS Properties

The and Tags

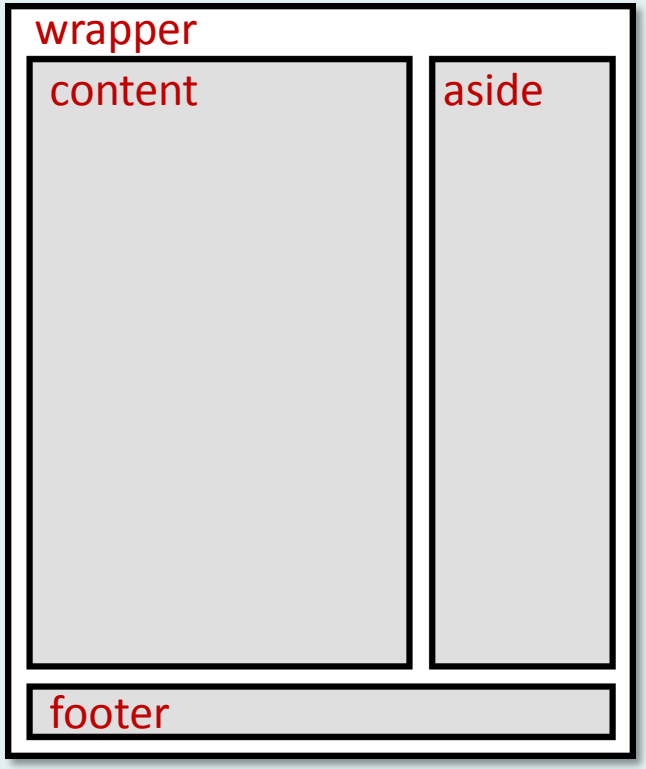
```
<div class="book">
<p>The novel Robinson Crusoe, written by
<span class="author">Daniel Defoe</span>,
was published in 1719.</p>
</div>
```

- No inherent formatting (they do nothing)
- Ideal for applying classes and ids
- The tag is an *inline* element
- The tag is used to define content sections, it is a *block-level* element

There is no <author> element in html but one can be created (sort of) using and an appropriate class name.

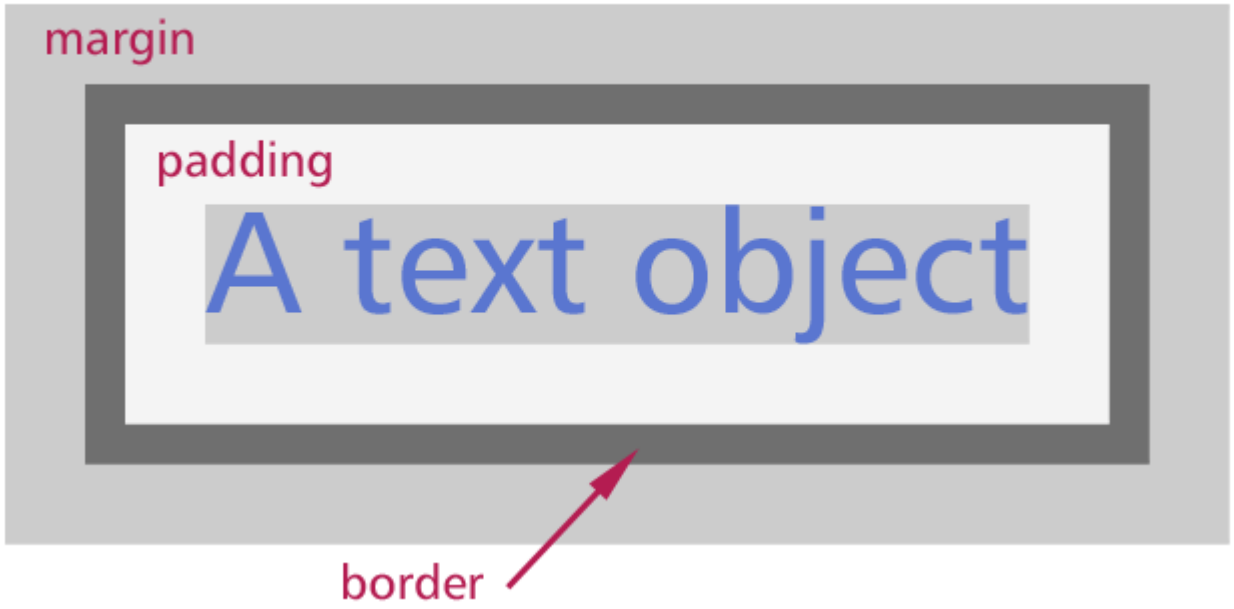
Page Structure with

```
<body>
<div id="wrapper">
    <div id="content">
        <h1>Page title</h1>
        <p>Some text...</p>
        <h2>Sub-heading</h2>
        <p>More text...</p>
    </div> <!-- close content -->
    <div id="aside">
        <h2>Sub-heading</h2>
        <p>More text...</p>
    </div> <!-- close aside -->
    <div id="footer">
        <p>More text...</p>
    </div> <!-- close footer -->
</div> <!-- close wrapper -->
</body>
```



In XHTML, the element is used to structure documents. This idea is extended in HTML5 with new structural elements such as <section>, <header> and <footer>.

The CSS “Box Model”

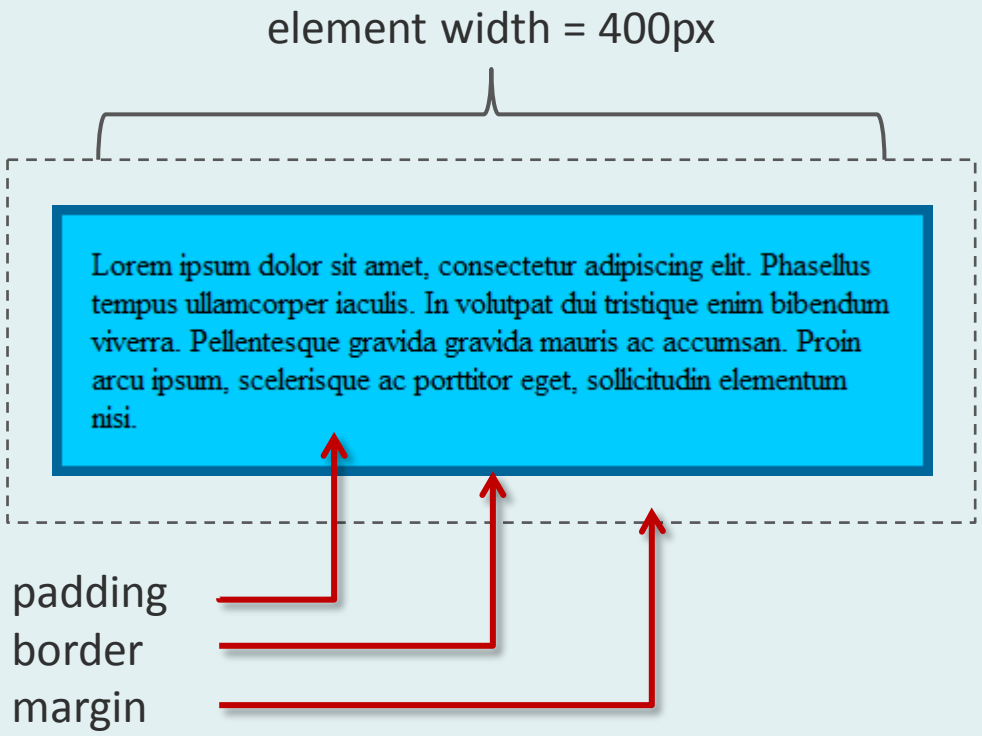


This box model applies to every HTML element, so you can think of every paragraph, heading or image as a box. In all cases, the padding, border and margin can be specified using CSS.

http://css-tricks.com/the-css-box-model/
http://www.hicksdesign.co.uk/boxmodel

Box Model Declarations

```
p {
   width: 400px;
   padding: 15px;
   border: 5px solid #069;
   margin: 20px;
   background-color:#0CF;
}
```



Element width does not include padding, border or margin, so in this case, the total width is 400 + 30 + 10 + 40 = 480px. Note that padding takes the element background colour, the border has an independent colour and the margin is transparent.

http://reference.sitepoint.com/css/boxmodel http://css-tricks.com/box-sizing/

Box-sizing in CSS3

The traditional CSS box model, which adds padding and border to the element width and height often causes confusion. Surely it would make more sense to have a model where the stated width of an element is fixed and the border and padding are subtracted from that width and not added to them. The result would be an element whose dimensions remain the same irrespective of the padding and border.

Well, you can now choose to have the box model behave in this way is you wish, using the CSS3 box-sizing property and setting the value to “border-box”:

```
.content {box-sizing: border-box;}
```

This sounds like a perfect solution and the only problem is that this property will not be recognised by older browsers but current support is good.

http://www.paulirish.com/2012/box-sizing-border-box-ftw/ http://caniuse.com/css3-boxsizing

Padding, Border and Margin

```
#content-main {
   margin: 20px;
}
```

```
#content-main {
   margin-top: 20px;
   margin-right: 20px;
   margin-bottom: 20px;
   margin-left: 20px;
}
```

```
#content-main {
   margin: 20px 20px 20px 20px;
}
```

All three rules on the left are equivalent – all four margins are set to 20px.

It is possible to set the width of padding, border and margin for the top, right, bottom and left of any element independently as in example 2.

Most often this is done using the short-hand notation (example 3) using the following order:
margin: *top right bottom left;*
begin at the top and move clockwise round the box.

Box Model Shorthand

```
#content-main {
   margin: 20px;
}

#content-main {
   margin: 20px 10px;
}

#content-main {
   margin: 20px 10px 40px;
}

#content-main {
   margin: 20px 15px 25px 40px;
}
```

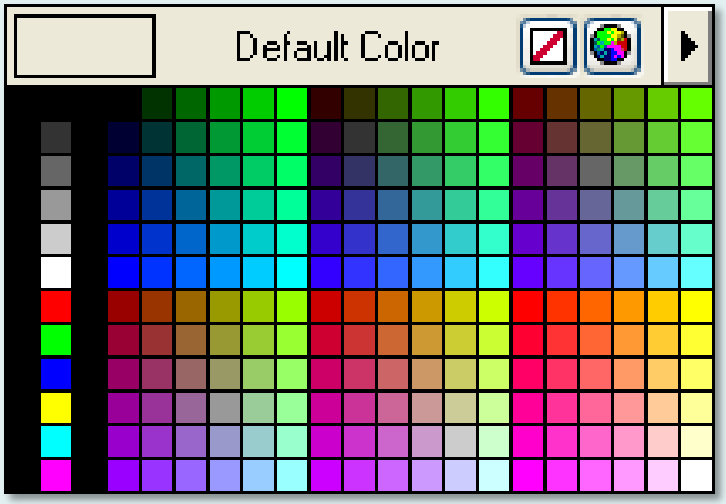
Declarations for padding and margin can be made using shorthand and all four examples on the left are valid.

Where only one value is specified, it applies to all sides. When two values are specified, the first applies to top and bottom and the second to left and right. For three values, the first is top, the second is left and right and the third is bottom. Four values are used to specify each side independently.

Colours and CSS

```
body {
    background-color: #8AC6CD;
}
```

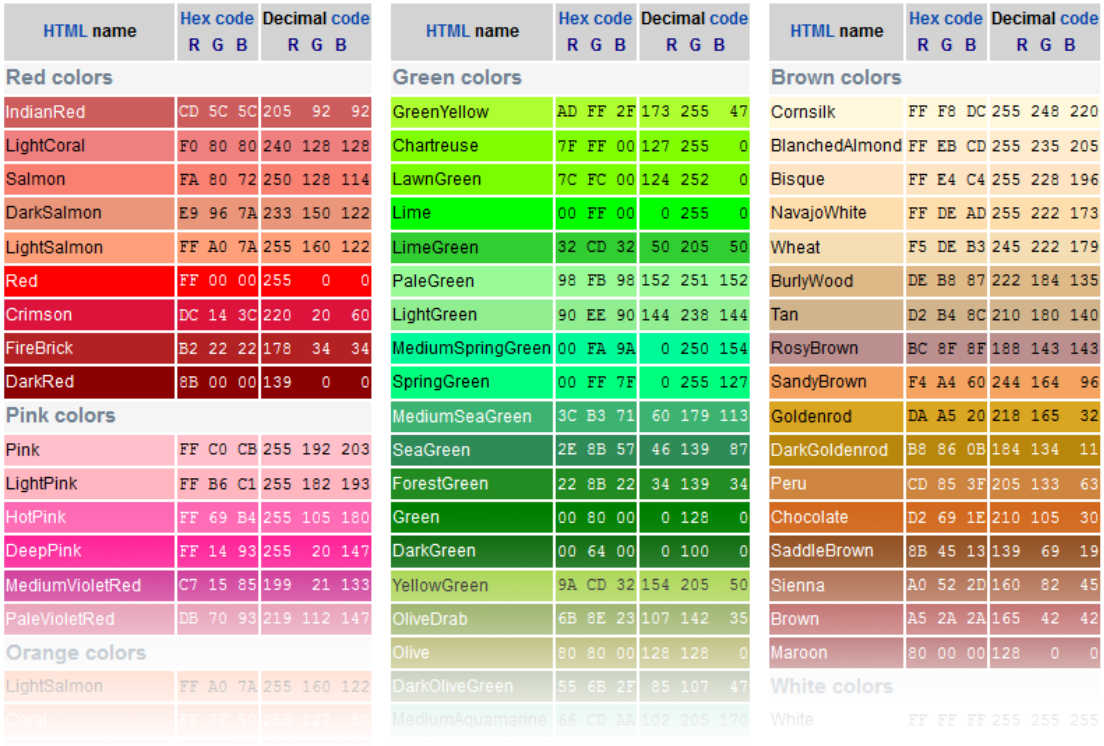
Web Safe Colours?



There are 216 colours in the “web safe” colour palette. This was devised to help web designers choose colours that would reproduce reliably on older hardware. These days, the vast majority of monitors can display “true colour” and the restricted palette is rarely considered – it is an historical curiosity.

You may use any one of the **16,777,216** colours available to you on your web sites (but not all at the same time!).

Colours by Name



HTML name	Hex code R G B	Decimal code R G B
Red colors		
IndianRed	CD 5C 5C	205 92 92
LightCoral	F0 80 80	240 128 128
Salmon	FA 80 72	250 128 114
DarkSalmon	E9 96 7A	233 150 122
LightSalmon	FF A0 7A	255 160 122
Red	FF 00 00	255 0 0
Crimson	DC 14 3C	220 20 60
FireBrick	B2 22 22	178 34 34
DarkRed	8B 00 00	139 0 0
Pink colors		
Pink	FF C0 CB	255 192 203
LightPink	FF B6 C1	255 182 193
HotPink	FF 69 B4	255 105 180
DeepPink	FF 14 93	255 20 147
MediumVioletRed	C7 15 85	199 21 133
PaleVioletRed	DB 70 93	219 112 147
Orange colors		
LightSalmon	FF A0 7A	255 160 122
Coral	FF 7F 50	255 127 80

HTML name	Hex code R G B	Decimal code R G B
Green colors		
GreenYellow	AD FF 2F	173 255 47
Chartreuse	7F FF 00	127 255 0
LawnGreen	7C FC 00	124 252 0
Lime	00 FF 00	0 255 0
LimeGreen	32 CD 32	50 205 50
PaleGreen	98 FB 98	152 251 152
LightGreen	90 EE 90	144 238 144
MediumSpringGreen	00 FA 9A	0 250 154
SpringGreen	00 FF 7F	0 255 127
MediumSeaGreen	3C B3 71	60 179 113
SeaGreen	2E 8B 57	46 139 87
ForestGreen	22 8B 22	34 139 34
Green	00 80 00	0 128 0
DarkGreen	00 64 00	0 100 0
YellowGreen	9A CD 32	154 205 50
OliveDrab	6B 8E 23	107 142 35
Olive	80 80 00	128 128 0
DarkOliveGreen	55 6B 2F	85 107 47
MediumAquamarine	66 CD AA	102 205 170

HTML name	Hex code R G B	Decimal code R G B
Brown colors		
Cornsilk	FF F8 DC	255 248 220
BlanchedAlmond	FF EB CD	255 235 205
Bisque	FF E4 C4	255 228 196
NavajoWhite	FF DE AD	255 222 173
Wheat	F5 DE B3	245 222 179
BurlyWood	DE B8 87	222 184 135
Tan	D2 B4 8C	210 180 140
RosyBrown	BC 8F 8F	188 143 143
SandyBrown	F4 A4 60	244 164 96
Goldenrod	DA A5 20	218 165 32
DarkGoldenrod	B8 86 0B	184 134 11
Peru	CD 85 3F	205 133 63
Chocolate	D2 69 1E	210 105 30
SaddleBrown	8B 45 13	139 69 19
Sienna	A0 52 2D	160 82 45
Brown	A5 2A 2A	165 42 42
Maroon	80 00 00	128 0 0
White colors		
White	FF FF FF	255 255 255

Colours in CSS may be specified by name and there are well over one hundred named colours. However, that's a small proportion of the **16 million colours** available to you in the RGB colour space. Consequently, colour names are rarely used and are considered an interesting curiosity.

Web colours

Specifying colours in CSS

Colours can be specified in the following ways:

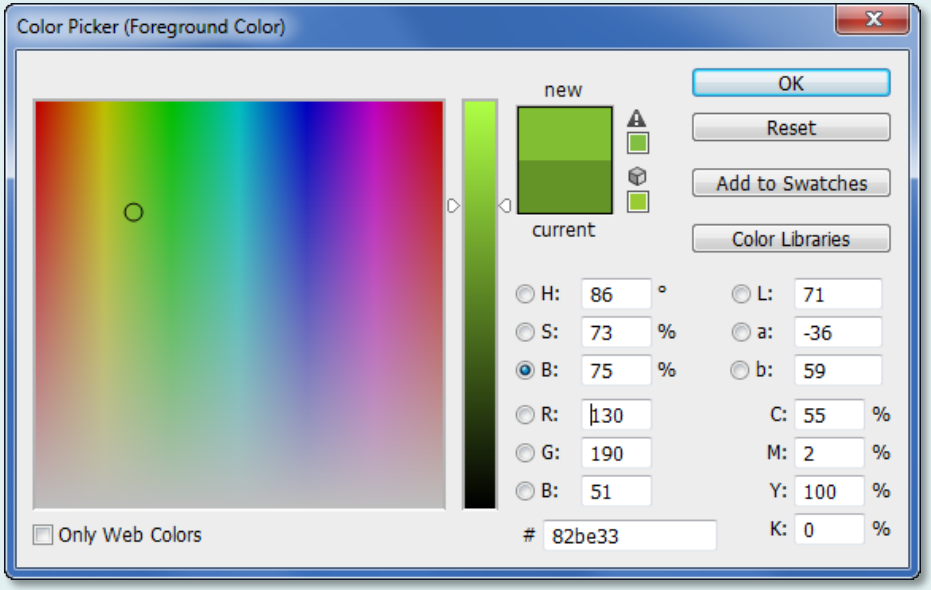
RGB	Red, Green, Blue	color: rgb(255,0,0)
Hex	Base 16 RGB	color: #FF0000
Hex shorthand		color: #F00

Additional CSS3 options

RGBA	RGB plus Alpha	color: rgba(255, 0, 0, 1)
HSL	Hue, Saturation, Lightness	color: hsl(0,100%, 50%)
HSLA	HSL plus Alpha	color: hsla(0,100%,50%,1)

Alpha specifies the degree of opacity in a colour in a range from 0 = transparent to 1 = Opaque. All the colours specified above are Red. Currently, hexadecimal is most common but RGBA tends to be used when changes to opacity are required.

Hexadecimal Colours



Hexadecimal (base 16) colours are based on the Red-Green-Blue (RGB) colour space. Any colour can be described using the varying amount of those 3 colours in a range from 0 to 255. Hexadecimal is simply a shorthand . Rather than using 3 decimal numbers (one for each colour), it uses 3 base 16 pairs with a “#” prefix.

The Photoshop colour picker can be used to select the colour you want and it will calculate the hexadecimal value for you.

Red: AC Green: D3 Blue: 73 = #ACD373

is the same as:

Red: 172 Green: 211 Blue: 115 in decimal

http://www.mathsisfun.com/hexadecimal-decimal-colors.html

Fonts and CSS

I like this font,
can I use it?

Web Safe Fonts?

Arial

Comic Sans MS

Georgia

Impact

Times New Roman

Trebuchet MS

Verdana

 (Webdings)

The standard Windows fonts

A browser can only display the fonts specified in CSS rules if that font is available on the client computer.*

Web safe fonts are those that we can be sure exist on the viewer's computer. For example, all the standard Windows or MacOS fonts that are installed with the operating system.

*CSS3 reintroduces the @fontface method that allows fonts to be displayed even if not stored locally but our focus is on CSS 2.1 for now.

http://www.ampsoft.net/webdesign-l/WindowsMacFonts.html

The font-family declaration

```
h1{
   font-family: Georgia, "Times New Roman", Times, serif;
   color: #534741;
   padding-bottom: 5px;
   border-bottom: 3px solid #DF7414;
}
```

The font-family declaration is used to specify fonts. Rather than specifying one font, it's good practice to specify a "font stack". If the first font is not available, the second will be used and so on. In the example above, Georgia is the preferred font but if it's not available, Times New Roman will be used. In general, a stack should have a minimum of 3 fonts; one for Windows, one for MacOS and a default generic font for anything else.

font-family

CSS Font Stack

Font Stacks

Although using the limited number of web safe fonts *may* ensure that every user gets to see the same typographic design, you can use the principle of progressive enhancement (or more accurately in this case, graceful degradation) to specify your preferred fonts to those clients who can display them, while relying on the fall-back of more reliable fonts in the font stack for those clients who can't.

For example, here's a font stack for Calibri:

```
font-family: Calibri, Candara, Segoe, "Segoe UI", Optima, Arial, sans-serif;
```

And here's another for Garamond:

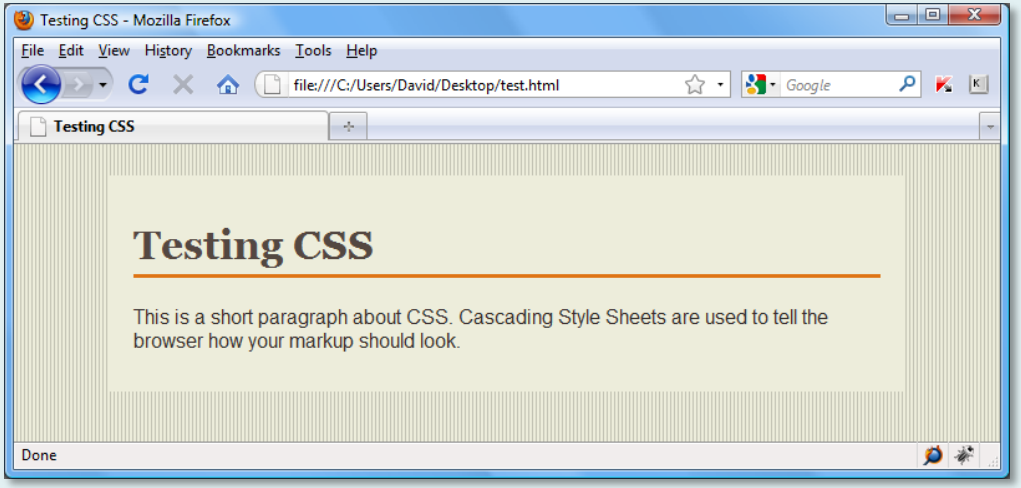
```
font-family: Garamond, Baskerville, "Baskerville Old Face", "Hoefler Text", "Times New Roman", serif;
```

Note that font names containing spaces must be quoted.

http://cssfontstack.com

A Worked Example

Putting CSS to Use

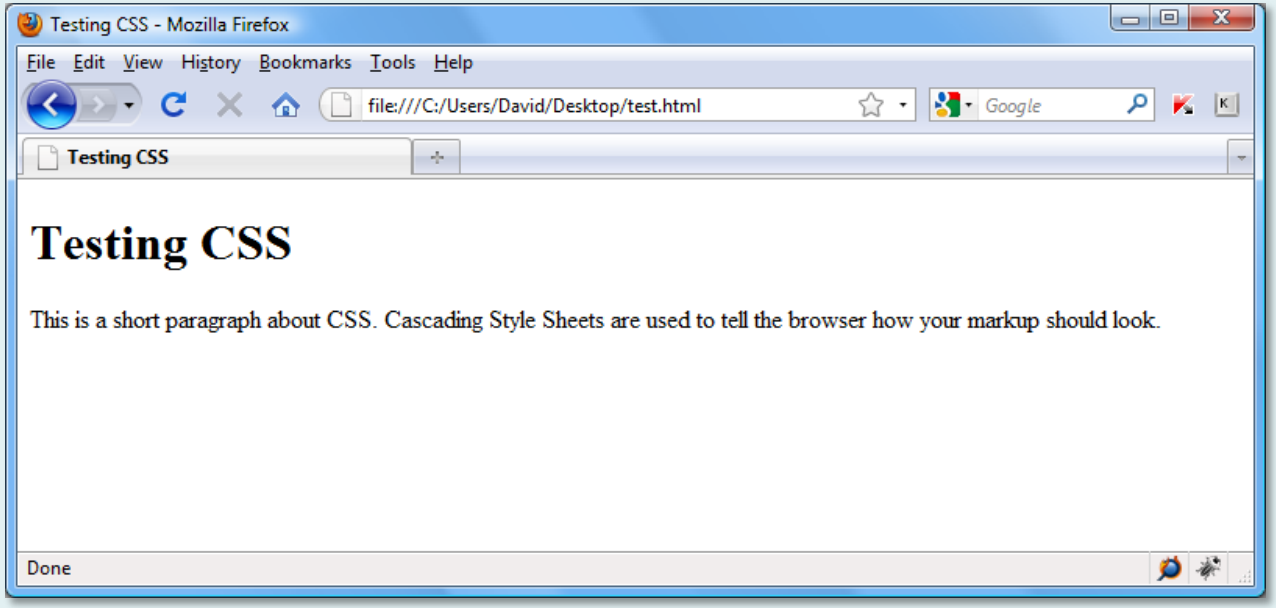


How to do this?

Working with CSS is relatively simple but remember that there are two text files involved, the HTML file and the CSS file. The CSS file is linked to the HTML file using <link>.

CSS in Action

```
<div id="content">
<h1>Testing CSS</h1>
<p>This is a short paragraph about CSS.
Cascading Style Sheets are used to tell the
browser how your markup should look.</p>
</div> <!-- close content -->
```

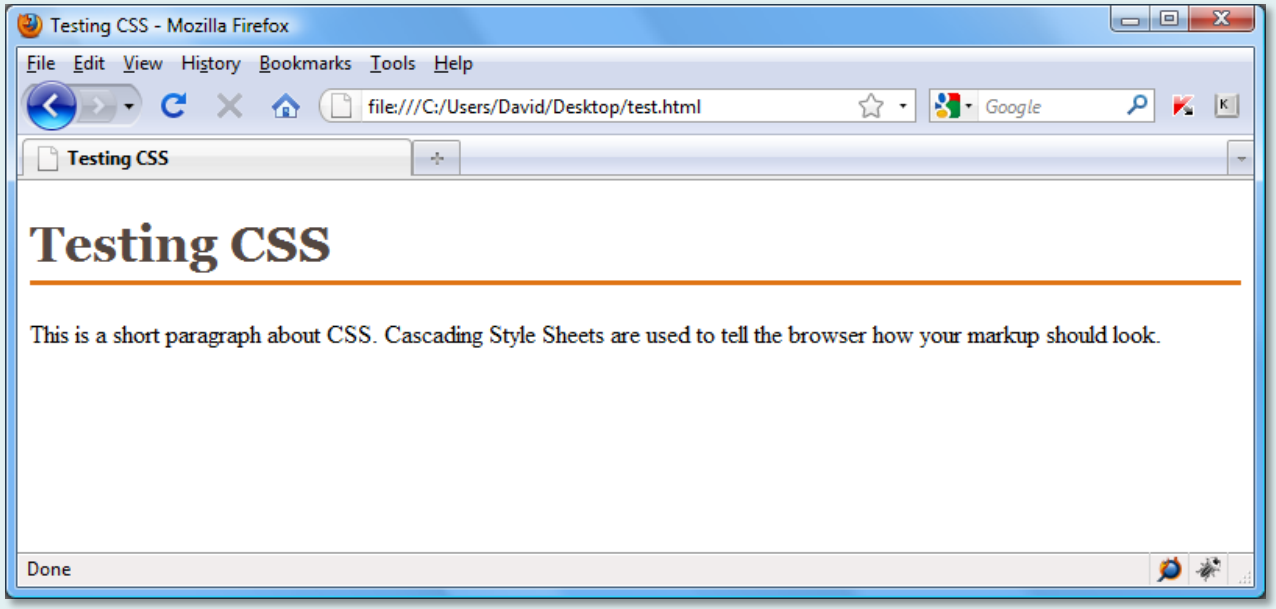


Without CSS, the browser displays markup using its own default styles. Typically, this will be black foreground, white background and Times New Roman text.

CSS in Action

```
<div id="content">
<h1>Testing CSS</h1>
<p>This is a short paragraph about CSS.
Cascading Style Sheets are used to tell the
browser how your markup should look.</p>
</div> <!-- close content -->
```

```
h1 {
font-family:Georgia, "Times New
Roman", Times, serif;
color:#534741;
padding-bottom:5px;
border-bottom:3px solid #DF7414;
}
```

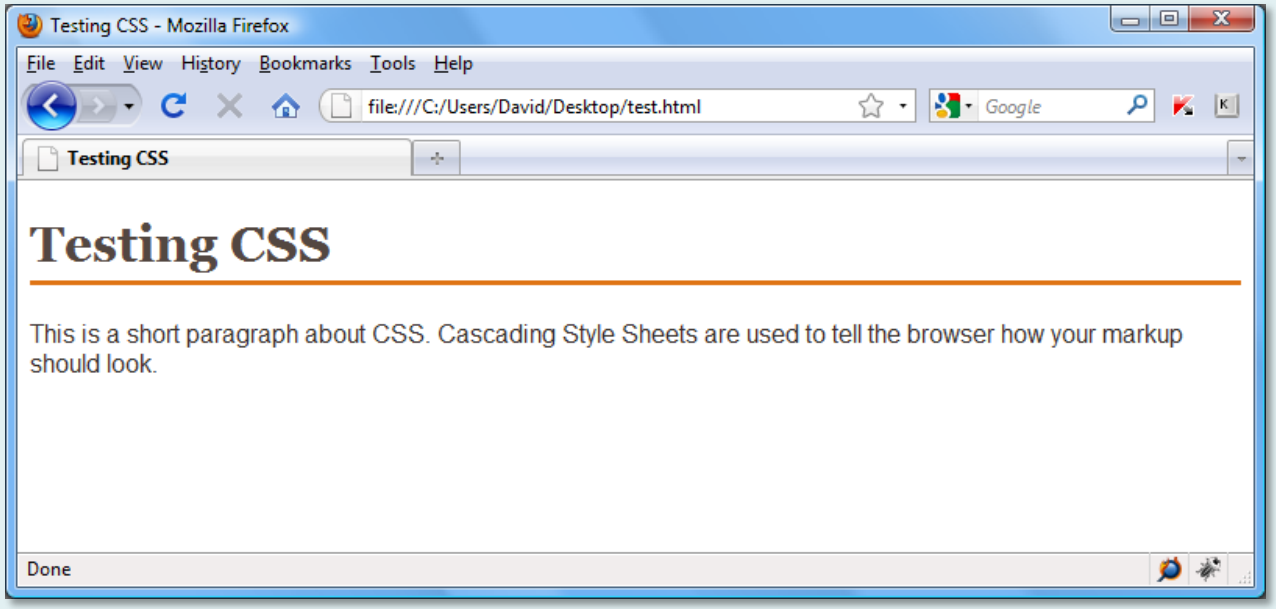


A CSS rule for the h1 element is added. This rule sets the font-family, the colour, the padding and the bottom border (underline). Note that colours are defined using hexadecimal.

CSS in Action

```
<div id="content">
<h1>Testing CSS</h1>
<p>This is a short paragraph about CSS.
Cascading Style Sheets are used to tell the
browser how your markup should look.</p>
</div> <!-- close content -->
```

```
p {
font-family: Arial, Helvetica, sans-
serif;
color: #362F2D;
}
```

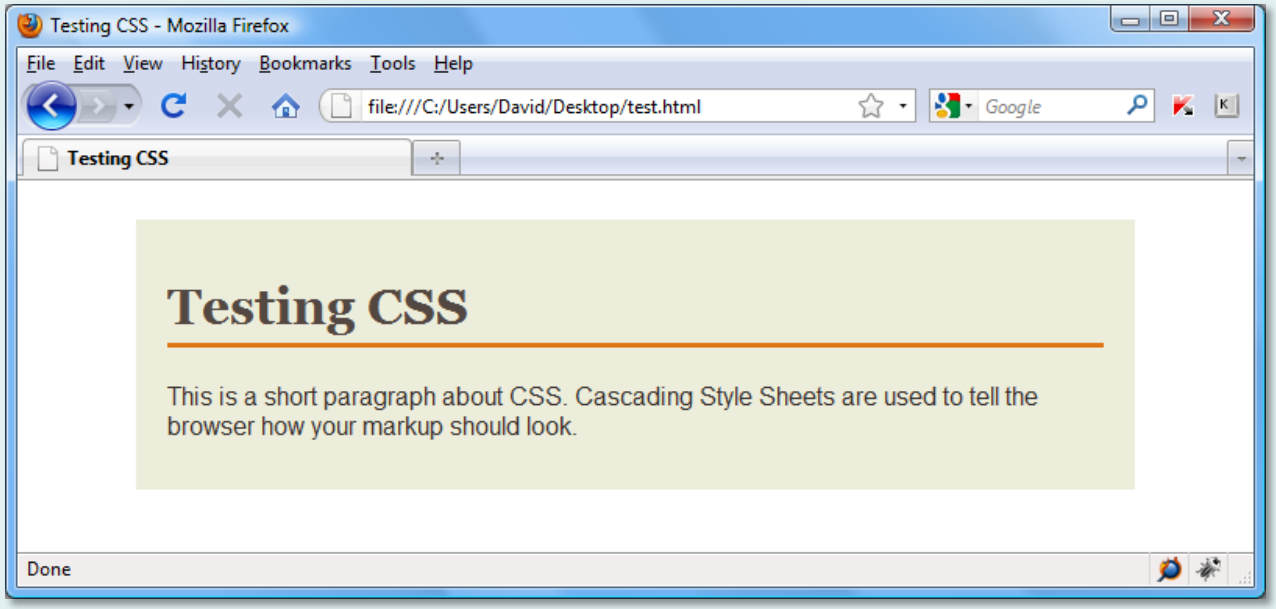


The two declarations which form the rule for <p> set the font-family and the colour. Note the use of different fonts, which form the “font stack”.

CSS in Action

```
<div id="content">
<h1>Testing CSS</h1>
<p>This is a short paragraph about CSS.
Cascading Style Sheets are used to tell the
browser how your markup should look.</p>
</div> <!-- close content -->
```

```
#content {
width: 600px;
padding: 15px 20px;
margin: 25px auto;
background-color: #EDEDDB;
}
```

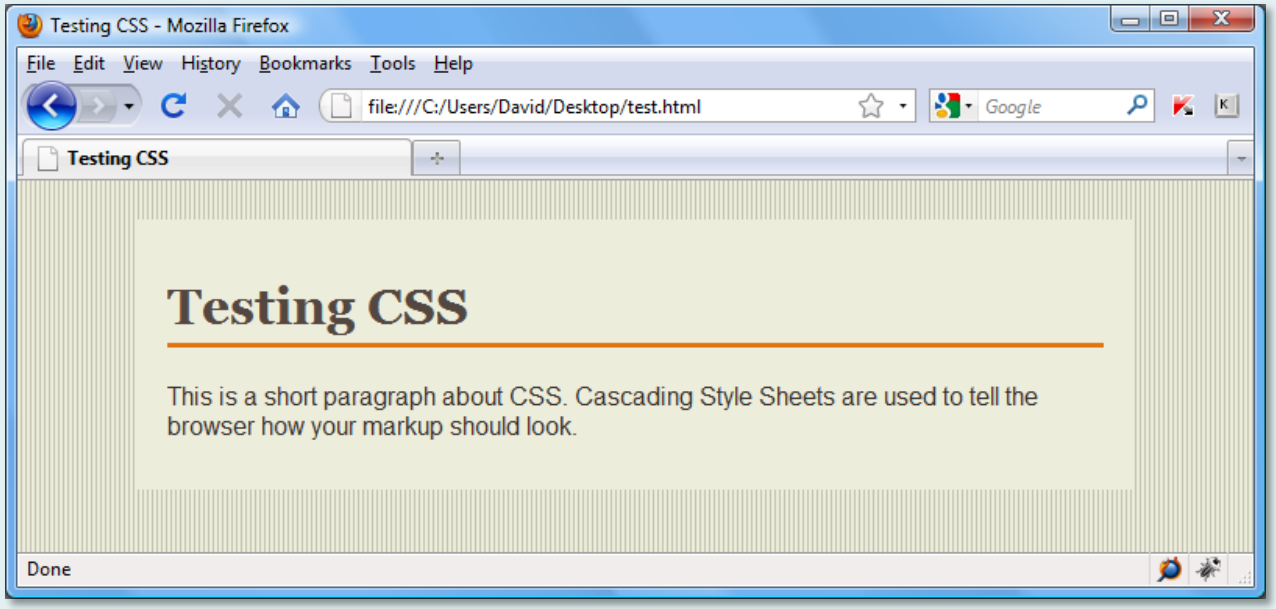


The next rule applies to the content . It sets a fixed width, some padding and a background colour. Left and right margins are set to “auto”, which has the effect of centring the div on the page. Note that the total width of our content box will be 640px (600+20+20).

CSS in Action

```
<div id="content">
<h1>Testing CSS</h1>
<p>This is a short paragraph about CSS.
Cascading Style Sheets are used to tell the
browser how your markup should look.</p>
</div> <!-- close content -->
```

```
body {
background-image:
url(background.png);
}
```



The rule for the <body> tag is used here to set the background. It uses a small .png image that repeats to form a pattern. Note that the background to the is not affected – it is in front of the body and has a background colour of its own.

The Finished Stylesheet

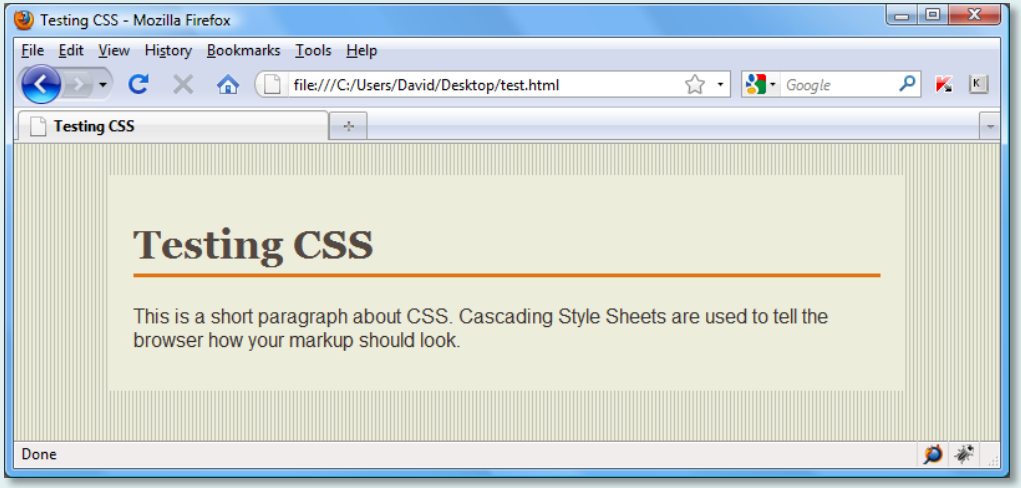
```
body {
background-image:url(background.png);
}

h1 {
font-family:Georgia, "Times New Roman", Times, serif;
color:#534741;
padding-bottom:5px;
border-bottom:3px solid #DF7414;
}

p {
font-family:Arial, Helvetica, sans-serif;
color:#362F2D;
}

#content {
width:600px;
padding:15px 20px;
margin:25px auto;
background-color:#EDEDDB;
}
```

Notice that the order of the CSS rules in the stylesheet are broadly hierarchical. This is not necessary but is good practice and helps locate rules in complex stylesheets.



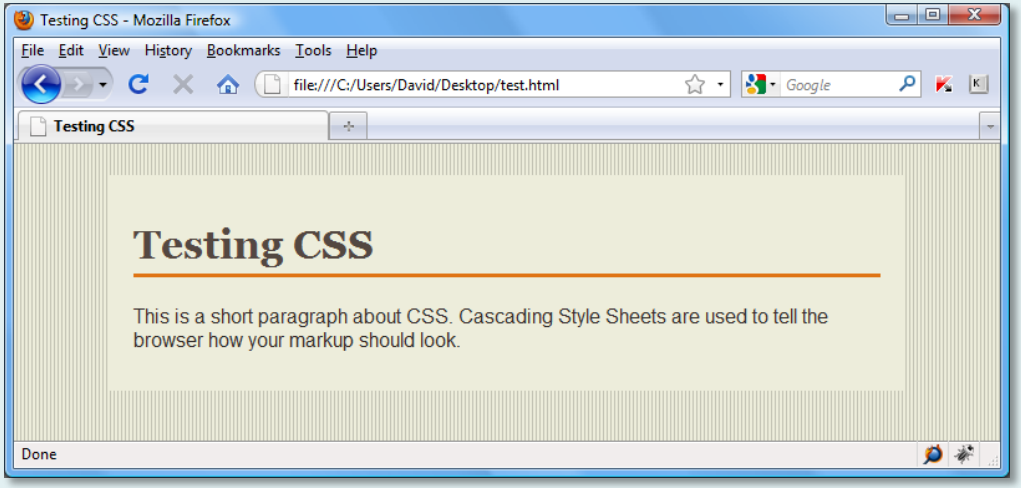
Stylesheet Comments

```
/* Global styles */
body {
background-image:url(background.png);
}

/* Typographic styles */
h1 {
font-family:Georgia, "Times New Roman", Times, serif;
color:#534741;
padding-bottom:5px;
border-bottom:3px solid #DF7414;
}
p {
font-family:Arial, Helvetica, sans-serif;
color:#362F2D;
}

/* Page structure */
#content {
width:600px;
padding:15px 20px;
margin:25px auto; /* centred */
background-color:#EDEDDB;
}
```

It is good practice to add explanatory comments to your CSS. This will help others interpret your rules and it will help you when you are developing your own designs.



CSS Inheritance

Some properties are inherited by child elements from their parent and others are not. In general, font and colour properties are inherited while box model properties (e.g. margin, border) are not.

For example, if you wanted the font to be the same on all text elements, you could add a style rule for the <body> and all elements within body would inherit that font.

```
/* all text to be Verdana */
   body {
   font-family: Verdana, Geneva, sans-serif;
   }
```

On the other hand, you wouldn't want padding inherited from <body>

```
/* only body gets padding */
   body {
   padding: 20px;
   }
```

Generally, inheritance works as you might expect.

http://reference.sitepoint.com/css/inheritance

Working with Inheritance

One common situation you may face is that you want all text on a page (paragraphs, lists, tables etc.) to display in one font except for the headings, which you want to display in a different font. Working with inheritance, we can easily achieve this outcome with just 2 style rules:

```
/* all text to be Verdana */
   body {
   font-family: Verdana, Geneva, sans-serif;
   }
/* except headings */
  h1, h2, h3, h4, h5, h6 {
  font-family: Georgia, Times, "Times New Roman", serif;
  }
```

The first rule sets all text to Verdana (including headings) but the second rule overrides the first for headings h1-h6. This feature of CSS is known as the cascade and the way it works depends on the order and specificity of the style rules. This can be a little complicated, so for now, let's keep things simple and just accept that it works for the example above.

http://reference.sitepoint.com/css/inheritancecascade

Overriding Browser Default Styles

CSS Resets

In the previous example, you will notice that we didn't bother setting all possible style options. For example, we didn't set the height for <h1>, we just accepted the browser default for that element.

```
h1 {
   font-family: Georgia, "Times New Roman", Times, serif;
   color: #534741;
   padding-bottom: 5px;
   border-bottom: 3px solid #DF7414;
}
```

In many cases and especially for simple websites, it may be fine to mix your own specific rules with the browser default rules. However, not all browsers use exactly the same defaults and this may lead to inconsistencies in your design.

CSS resets are a way of setting all elements to a baseline value that will be true in all browsers. The upside of this approach is consistency, the downside is you need to add more rules.

Resetting your styles with CSS reset

How do I reset CSS?

There are a number of different methods for resetting CSS, some minimal, some more complicated. Most web designers will use one of the off-the-shelf CSS resets. The method you choose will depend on the type of site you are building and personal preference.

```
/* the universal selector reset */
  * {
    margin: 0;
    padding: 0;
  }
```

The universal selector reset is probably the simplest method. This works on the principle that the asterisk character acts as a wildcard and represents all elements. In effect the margin and padding on all elements is set to zero. This might be OK for simple sites but it's a crude approach and may cause problems down the line.

http://www.cssreset.com/scripts/universal-selector-css-reset/

```
/* http://meyerweb.com/eric/tools/css/reset/
   v2.0 | 20110126
   License: none (public domain)
*/

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
                        margin: 0;
                        padding: 0;
                        border: 0;
                        font-size: 100%;
                        font: inherit;
                        vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
                        display: block;
}
body {
                        line-height: 1;
}
ol, ul {
                        list-style: none;
}
blockquote, q {
                        quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
                        content: '';
                        content: none;
}
table {
                        border-collapse: collapse;
                        border-spacing: 0;
}
```

Eric Meyer's Reset CSS

The most popular reset in use today is Eric Meyer's Reset CSS. Rather than taking the nuclear option as the universal selector reset does, it is designed to reset only those elements that will benefit from resetting and leaving others alone.

Version 2.0 of the reset adds some extra rules for HTML5.

Eric Meyer's Reset CSS

http://www.cssreset.com/

How do I apply a reset?

There are other popular reset methods, such as the HTML5 Doctor CSS Reset (an evolution of the Eric Myer reset) but whichever method you choose (assuming you want to use a reset at all – it's not compulsory), you should either add the reset to the top of your main CSS file, before your own rules or put the reset in a separate CSS file (reset.css) and link to it from the head of your HTML file before linking any other CSS.

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

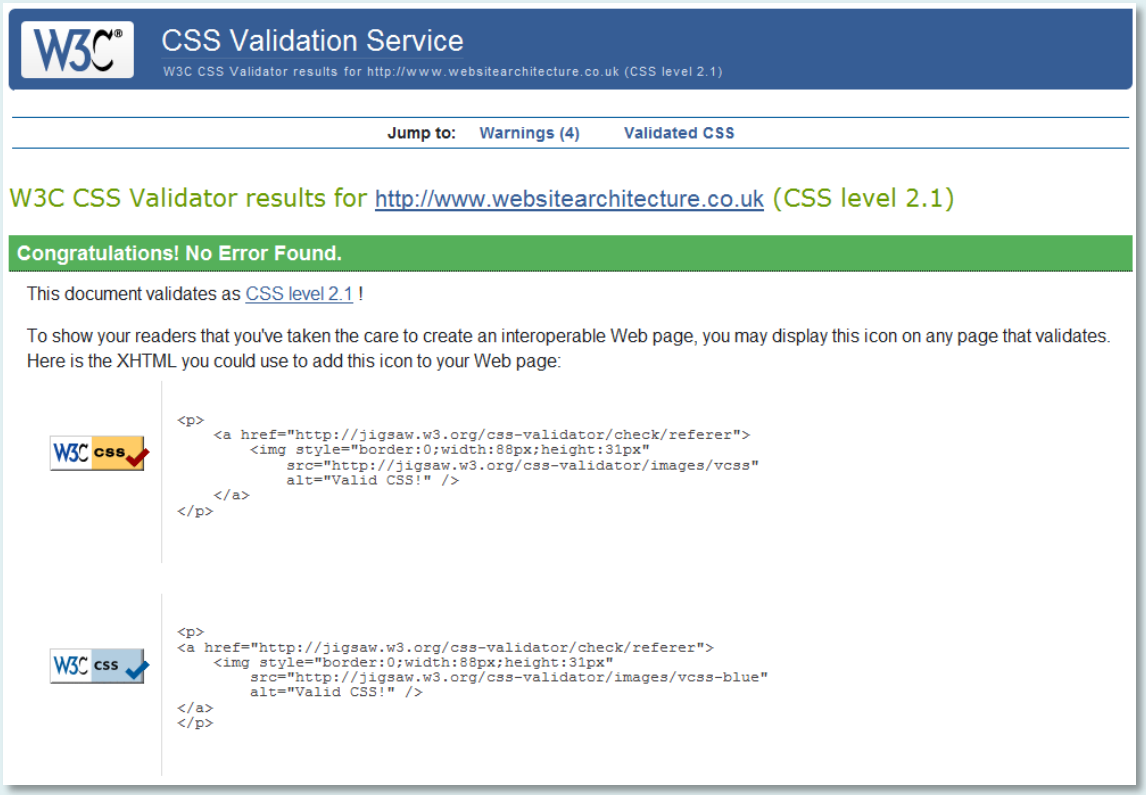
In HTML5, this becomes:

```
<link rel="stylesheet" href="reset.css">
<link rel="stylesheet" href="main.css">
```

We don't need the type attribute in HTML5 (it's implicit) and we don't need to close the link element.

HTML5 Doctor CSS Reset

Validate



CSS Validation Service

W3C CSS Validator results for http://www.websitearchitecture.co.uk (CSS level 2.1)

Jump to: Warnings (4) Validated CSS

W3C CSS Validator results for http://www.websitearchitecture.co.uk (CSS level 2.1)

Congratulations! No Error Found.

This document validates as CSS level 2.1 !

To show your readers that you've taken the care to create an interoperable Web page, you may display this icon on any page that validates. Here is the XHTML you could use to add this icon to your Web page:

```
<p>
    <a href="http://jigsaw.w3.org/css-validator/check/referer">
        <img style="border:0;width:88px;height:31px"
            src="http://jigsaw.w3.org/css-validator/images/vcss"
            alt="Valid CSS!" />
    </a>
</p>
```

```
<p>
<a href="http://jigsaw.w3.org/css-validator/check/referer">
    <img style="border:0;width:88px;height:31px"
        src="http://jigsaw.w3.org/css-validator/images/vcss-blue"
        alt="Valid CSS!" />
</a>
</p>
```

CSS can be validated in the same way as XHTML. Note that we are using CSS level 2.1 in this case.

http://jigsaw.w3.org/css-validator/

The End