CSS - cascade style sheet

CSS works by associating rules with HTML elements. These rules govern how the content of specified elements should be displayed. A CSS rule contains two parts: a selector and a declaration.

**Syntax:-**



Selectors indicate which element the rule applies to. The same rule can apply to more than one element if you separate the element names with commas.

Declarations indicate how the elements referred to in the selector should be styled. Declarations are split into two parts (a property and a value), and are separated by a colon.

CSS declarations sit inside curly brackets and each is made up of two parts: a property and a value, separated by a colon. You can specify several properties in one declaration, each separated by a semi-colon.



Properties indicate the aspects of the element you want to change. For example, color, font, width, height and border.

Values specify the settings you want to use for the chosen properties. For example,if want to specify a color property then the value is the color you want the text in these elements to be.

USING EXTERNAL CSS

###### <link>

The **<link>** element can be used in an HTML document to tell the browser where to find the CSS file used to style the page. It is an empty element (meaning it does not need a closing tag), and it lives inside the **<head>** element. It should use three attributes:

**href**

This specifies the path to the CSS file (which is often placed in a folder called **css** or **styles)**.

**type**

This attribute specifies the type of document being linked to. The value should be **text/css**.

**rel**

This specifies the relationship between the HTML page and the file it is linked to. The value should be **stylesheet** when linking to a CSS file.

USING INTERNAL CSS

<style>

You can also include CSS rules within an HTML page by placing them inside a **<style>** element, which usually sits inside the **<head>** element of the page.

The **<style>** element should use the **type** attribute to indicate that the styles are specified in CSS. The value should be **text/ css**.

When building a site with more than one page, you should use an external CSS style sheet. This:

* Allows all pages to use the same style rules (rather than repeating them in each page).
* Keeps the content separate from how the page looks.
* Means you can change the styles used across all pages by altering just one file (rather than each individual page).

##### CSS RULE CASCADE

##### If there are two or more rules that apply to the same element, it is important to understand which will take precedence.

LAST RULE

If the two selectors are identical, the latter of the two will take precedence. Here you can see the second **i** selector takes precedence over the first.

SPECIFICITY

If one selector is more specific than the others, the more specific rule will take precedence over more general ones. In this example:

**h1** is more specific than **\* p b** is more specific than **p p#intro** is more specific than **p**

IMPORTANT

You can add **!important** after any property value to indicate that it should be considered more important than other rules that apply to the same element.

##### CSS SELECTORS

##### There are many different types of CSS selector that allow you to target rules to specific elements in an HTML document.CSS selectors are case sensitive, so they must match element names and attribute values exactly.

|  |  |  |
| --- | --- | --- |
| SELECTOR | MEANING | EXAMPLE |
| UNIVERSAL SELECTOR | Applies to all elements in the | **\* {}** |
|  | document | Targets all elements on the page |
| TYPE SELECTOR | Matches element names | **h1, h2, h3 {}** |
|  |  | Targets the **<h1>**, **<h2>** and **<h3>** |
|  |  | elements |
| CLASS SELECTOR | Matches an element whose | **.note {}** |
|  | **class** attribute has a value that matches the one specified after the period (or full stop) symbol | Targets any element whose **class**  attribute has a value of **note**  **p.note {}** |
|  |  | Targets only **<p>** elements |
|  |  | whose **class** attribute has a |
|  |  | value of **note** |
| ID SELECTOR | Matches an element whose | **#introduction {}** |
|  | **id** attribute has a value that matches the one specified after | Targets the element whose  **id** attribute has a value of |
|  | the pound or hash symbol | **introduction** |
| CHILD SELECTOR | Matches an element that is a | **li>a {}** |
|  | direct child of another | Targets any **<a>** elements that |
|  |  | are children of an **<li>** element |
|  |  | (but not other **<a>** elements in |
|  |  | the page) |
| DESCENDANT SELECTOR | Matches an element that is a | **p a {}** |
|  | descendent of another specified | Targets any **<a>** elements that |
|  | element (not just a direct child of | sit inside a **<p>** element, even if |
|  | that element) | there are other elements nested |
|  |  | between them |
| ADJACENT SIBLING | Matches an element that is the | **h1+p {}** |
| SELECTOR | next sibling of another | Targets the first **<p>** element |
|  |  | after any **<h1>** element (but not |
|  |  | other **<p>** elements) |
| GENERAL SIBLING | Matches an element that is a | **h1~p {}** |
| SELECTOR | sibling of another, although it | If you had two **<p>** elements that |
|  | does not have to be the directly | are siblings of an **<h1>** element, |
|  | preceding element | this rule would apply to both |

##### INHERITANCE

* If you specify the **font-family** or **color** properties on the **<body>** element, they will apply to most child elements. This is because the value of the **font-family** property is **inherited** by child elements. It saves you from having to apply these properties to as many elements (and results in simpler style sheets).
* You can compare this with the **background-color** or **border** properties; they are **not inherited** by child elements. If these were inherited by all child elements then the page could look quite messy.
* You can force a lot of properties to inherit values from their parent elements by using **inherit** for the value of the properties.

#### COLOR

The **color** property allows you to specify the color of text inside an element. You can specify any color in CSS in one of three ways:

RGB VALUES

These express colors in terms of how much red, green and blue are used to make it up. For example:

**rgb(100,100,90)**

HEX CODES

These are six-digit codes that represent the amount of red, green and blue in a color, preceded by a pound or hash # sign.

For example: **#ee3e80**

COLOR NAMES

There are 147 predefined color names that are recognized by browsers.

For example: DarkCyanhat are recognized by browsers.

#### Background-color

CSS treats each HTML element as if it appears in a box, and the **background-color** property sets the color of the background for that box.

You can specify your choice of background color in the same three ways you can specify foreground colors: RGB values, hex codes, and color names (covered on the next page).

If you do not specify a background color, then the background is transparent.

By default, most browser windows have a white background, but browser users can set a background color for their windows, so if you want to be sure that the background is white you can use the **background-color** property on the **<body>** element.

UNDERSTANDING COLOR

RGB VALUES HEX CODES

Values for red, green, and blue are expressed Hex values represent values for red, green, and

as numbers between 0 and 255. blue in hexadecimal code.

**rgb(102,205,170) #66cdaa**

This color is made up of the following values: The value of the red, 102, is

102 red expressed as 66 in hexadecimal

205 green code. The 205 of the green is

170 blue expressed as **cd** and the 170 of

the blue equates to **aa**.

HUE



Hue is near to the colloquial idea of color. Technically speaking however, a color can also have saturation and brightness as well as hue.

COLOR NAMES

Colors are represented by predefined names.

However, they are very limited in number.

**MediumAquaMarine**

There are 147 color names supported by browsers (this color is **MediumAquaMarine**). Most consider this to be a limited color palette, and it is hard to remember the name for each of the colors so (apart from white and black) they are not commonly used.

SATURATION BRIGHTNESS

Saturation refers to the amount of gray in a color. At Brightness (or "value") refers to how much black is

maximum saturation, there would be no gray in the in a color. At maximum brightness, there would be no

color. At minimum saturation, the color would be mostly black in the color. At minimum brightness, the color

gray. would be very dark.



OPACITY

CSS3 introduces the **opacity** property which allows you to specify the opacity of an element and any of its child elements.

The value is a number between **0.0** and **1.0** (so a value of **0.5** is 50% opacity and **0.15** is 15% opacity).

The CSS3 **rgba** property allows you to specify a color, just like you would with an RGB value, but adds a fourth value to indicate opacity. This value is known as an **alpha** value and is a number between **0.0** and **1.0** (so a value of **0.5** is 50% opacity and **0.15** is 15% opacity). The **rgba** value will only affect the element on which it is applied (not child elements).

Because some browsers will not recognize RGBA colors, you can offer a fallback so that they display a solid color. If there are two rules that apply to the same element, the latter of the two will take priority. To create the fallback, you can specify a color as a hex code, color name or RGB value, followed by the rule that specifies an RGBA value. If the browser understands RGBA colors it will use that rule. If it doesn't, it will use the RGB value.

HSL COLOR

CSS3 introduces an entirely new and intuitive way to specify colors using hue, saturation, and lightness values.

HUE

Hue is the colloquial idea of color. In HSL colors, hue is often represented as a color circle where the angle represents the color, although it may also be shown as a slider with values from 0 to 360.



SATURATION

Saturation is the amount of gray in a color. Saturation is represented as a percentage. 100% is full saturation and 0% is a shade of gray.



LIGHTNESS

Lightness is the amount of white (lightness) or black (darkness) in a color. Lightness is represented as a percentage. 0% lightness is black, 100% lightness is white, and 50% lightness is normal. Lightness is sometimes referred to as *luminosity*.



HSL & HSLA

The **hsl** color property has been introduced in CSS3 as an alternative way to specify colors. The value of the property starts with the letters **hsl**, followed by individual values inside parentheses for:

HUE

This is expressed as an angle (between 0 and 360 degrees).

SATURATION

This is expressed as a percentage.

LIGHTNESS

This is expressed as a percentage with 0% being white, 50% being normal, and 100% being black.

The **hsla** color property allows you to specify color properties using hue, saturation, and lightness as above, and adds a fourth value which represents transparency (just like the **rgba** property). The **a** stands for:

ALPHA

This is expressed as a number between **0** and **1.0**. For example, **0.5** represents 50% transparency, and **0.75** represents 75% transparency.

#### TEXT

TYPEFACE TERMINOLOGY

SERIF

Serif fonts have extra details on the ends of the main strokes of the letters. These details are known as serifs.

Im

In print, serif fonts were traditionally used for long passages of text because they were considered easier to read.

SANS-SERIF

Sans-serif fonts have straight ends to letters, and therefore have a much cleaner design.

Im

Screens have a lower resolution than print. So, if the text is small, sans-serif fonts can be clearer to read.

MONOSPACE

Every letter in a monospace (or fixed-width) font is the same width. (Non-monospace fonts have different widths.)

im

Monospace fonts are commonly used for code because they align nicely, making the text easier to follow.

WEIGHT

Light

Medium

**Bold**

**Black**

The font weight not only adds emphasis but can also affect the amount of white space and contrast on a page.

STYLE

Normal

*Italic*

*Oblique*

Italic fonts have a cursive aspect to some of the lettering. Oblique font styles take the normal style and put it on an angle.

STRETCH

Condensed

Regular

Extended

In condensed (or narrow) versions of the font, letters are thinner and closer together. In expanded versions they are thicker and further apart.

SPECIFYING TYPEFACES

###### font-family

The **font-family** property allows you to specify the typeface that should be used for any text inside the element(s) to which a CSS rule applies.

The value of this property is the name of the typeface you want to use.

You can specify a list of fonts separated by commas so that, if the user does not have your first choice of typeface installed, the browser can try to use an alternative font from the list.

It is also common to end with a generic font name for that type of font.

If a font name is made up of more than one word, it should be put in double quote.

SIZE OF TYPE

###### font-size

The **font-size** property enables you to specify a size for the font. There are several ways to specify the size of a font. The most common are:

PIXELS

Pixels are commonly used because they allow web designers very precise control over how much space their text takes up. The number of pixels is followed by the letters **px**.

PERCENTAGES

The default size of text in browsers is 16px. So a size of 75% would be the equivalent of 12px, and 200% would be 32px.

If you create a rule to make all text inside the **<body>** element to be 75% of the default size (to make it 12px), and then specify another rule that indicates the content of an element inside the **<body>** element should be 75% size, it will be 9px (75% of the 12px font size).

EMS

An em is equivalent to the width of a letter m.

##### TYPE SCALES

8pt

9pt

10pt

11pt

12pt

14pt

18pt

24pt

36pt

48pt

60pt

72pt

##### UNITS OF TYPE SIZE

PIXELS PERCENTAGES EMS

TWELVE PIXEL SCALE

=

**h1 h2 h3**

**body**

**24px 18px 14px**

**12px**

**h1 h2 h3**

**body**

**200%**

**150%**

**117%**

**75%**

**h1 1.5em**

**h2 1.3em**

**h3 1.17em**

**body 100%**

**p 0.75em**

SIXTEEN PIXEL SCALE

=

**h1 h2 h3**

**body**

**32px 24px 18px**

**16px**

**h1 h2 h3**

**body**

**200%**

**150%**

**133%**

**100%**

**h1 2em**

**h2 1.5em**

**h3 1.125em**

**body 100%**

**p 1em**

PIXELS

Setting font size in pixels is the best way to ensure that the type appears at the size you intended (because percentages and ems are more likely to vary if a user has changed the default size of text in their browser).

Pixels are relative to the resolution of the screen, so the same type size will look larger when a screen has a resolution of 800x600 than it would when it is 1280x800.

You can also use **pt** for point sizes instead of **px** for pixels, but you should only do this when creating style sheets for printer- friendly versions of pages.

PERCENTAGES

The default size of text in a web browser is 16 pixels. Using percentages of this amount, you can create a scale where the default text size is 12 pixels, and headings are sized in relation to this.

It is possible for users to change the default size of text in their web browsers. If they have done this, the fonts will be displayed at the same scale that the designer intended, but at a larger size.

EMS

Ems allow you to change the size of text relative to the size of the text in the parent element. Since the default size of text in web browsers is 16 pixels, you can use similar rules to those shown for percentages.

Because users can change the default size of text in their browser, the fonts could all appear larger (or smaller) than the designer intended.

The extra **p** rule above is to help Internet Explorer 6 and 7 display the fonts at the right size. Without this extra rule, IE6 and IE7 exaggerate the relative sizes of other text.

###### @font-face

**@font-face** allows you to use a font, even if it is not installed on the computer of the person browsing, by allowing you to specify a path to a copy of the font, which will be downloaded if it is not on the user's machine.

Because this technique allows a version of the font to be downloaded to the user’s computer, it is important that the license for the font permits it to be used in this way.

**font-family**

This specifies the name of the font. This name can then be used as a value of the **font-family** property in the rest of the style sheet (as shown in the rule for the **<h1>** and **<h2>** elements).

**src**

This specifies the path to the font. In order for this technique to work in all browsers, you will probably need to specify paths to a few different versions of the font, as shown on the next page.

**format**

This specifies the format that the font is supplied in.

BOLD

###### font-weight

The **font-weight** property allows you to create bold text. There are two values that this property commonly takes:

**Normal**

This causes text to appear at a normal weight.

**bold**

This causes text to appear bold.

ITALIC

###### font-style

If you want to create italic text, you can use the **font-style** property. There are three values this property can take:

**normal**

This causes text to appear in a normal style (as opposed to italic or oblique).

**italic**

This causes text to appear italic.

**oblique**

This causes text to appear oblique.

UPPERCASE & LOWERCASE

###### text-transform

The **text-transform** property is used to change the case of text giving it one of the following values:

**uppercase**

This causes the text to appear uppercase.

**lowercase**

This causes the text to appear lowercase.

**capitalize**

This causes the first letter of each word to appear capitalized.

UNDERLINE & STRIKE

###### text-decoration

The **text-decoration** property allows you to specify the following values:

**none**

This removes any decoration already applied to the text.

**underline**

This adds a line underneath the text.

**overline**

This adds a line over the top of the text.

**line-through**

This adds a line through words.

**blink**

This animates the text to make it flash on and off (however this is generally frowned upon, as it is considered rather annoying).

##### LEADING

###### line-height

Leading (pronounced *ledding*) is a term typographers use for the vertical space between lines of text. In a typeface, the part of a letter that drops beneath the baseline is called a **descender**, while the highest point of a letter is called the **ascender**. Leading is measured from the bottom of the descender on one line to the top of the ascender on the next.



In CSS, the **line-height** property sets the height of an entire line of text, so the difference between the **font- size** and the **line-height** is equivalent to the leading (as shown in the diagram above).

Increasing the **line-height** makes the vertical gap between lines of text larger.

LETTER & WORD SPACING

###### letter-spacing, word-spacing

**Kerning** is the term typographers use for the space between each letter. You can control the space between each letter with the **letter-spacing** property.

It is particularly helpful to increase the kerning when your heading or sentence is all in uppercase. If your text is in sentence (or normal) case, increasing or decreasing the kerning can make it harder to read.

You can also control the gap between words using the **word-spacing** property.

When you specify a value for these properties, it should be given in ems, and it will be added on top of the default value specified by the font.

ALIGNMENT

###### text-align

The **text-align** property allows you to control the alignment of text. The property can take one of four values:

**left**

This indicates that the text should be left-aligned.

**right**

This indicates that the text should be right-aligned.

**center**

This allows you to center text.

**justify**

This indicates that every line in a paragraph, except the last line, should be set to take up the full width of the containing box.

VERTICAL ALIGNMENT

###### vertical-align

The **vertical-align** property is a common source of confusion. It is **not** intended to allow you to vertically align text in the middle of block level elements such as **<p>** and **<div>**, although it does have this effect when used with table cells (the **<td>** and **<th>** elements).

It is more commonly used with inline elements such as **<img>**, **<em>**, or **<strong>** elements. When used with these elements, it performs a task very similar to the HTML **align** attribute used on the **<img>** element, which you met on pages 103-106. The values it can take are:

**baseline**

**sub**

**super**

**top**

**text-top**

**middle**

**bottom**

**text-bottom**

It can also take a **length** (usually specified in pixels or ems) or a **percentage** of the line height.

INDENTING TEXT

###### text-indent

The **text-indent** property allows you to indent the first line of text within an element. The amount you want the line indented by can be specified in a number of ways but is usually given in pixels or ems.

It can take a negative value, which means it can be used to push text off the browser window. You can see this

technique used in this example, where the **<h1>** element uses a background image to represent the heading. The text has been moved far to the left, off the screen.

We still want the heading text to be on the page (for search engines and those who cannot see the image), but we cannot have it displayed on top of the logo or it will be unreadable. By pushing it 9,999 pixels to the left, it is way out of sight but still in the HTML code.

The second rule in this example indents the credits 20 pixels to the right.

##### DROP SHADOW

###### text-shadow

The **text-shadow** property has become commonly used despite lacking support in all browsers.

It is used to create a drop shadow, which is a dark version of the word just behind it and slightly offset. It can also be used to create an embossed effect by adding a shadow that is slightly lighter than the text.

The value of this property is quite complicated because it can take three lengths and a color for the drop shadow.

The first length indicates how far to the left or right the shadow should fall.

The second value indicates the distance to the top or bottom that the shadow should fall.

The third value is optional and specifies the amount of blur that should be applied to the drop shadow.

The fourth value is the color of the drop shadow.

##### FIRST LETTER OR LINE

###### : first - letter , : first – line

You can specify different values for the first letter or first line of text inside an element using

**:first-letter** and

**:first-line**.

Technically these are not properties. They are known as **pseudo-elements**.

You specify the pseudo-element at the end of the selector, and then specify the declarations as you would normally for any other element.

STYLING LINKS

###### : link , : visited

Browsers tend to show links in blue with an underline by default, and they will change the color of links that have been visited to help users know which pages they have been to.

In CSS, there are two **pseudo- classes** that allow you to set different styles for links that have and have not yet been visited.

**:link**

This allows you to set styles for links that have not yet been visited.

**:visited**

This allows you to set styles for links that have been clicked on.

They are commonly used to control colors of the links and also whether they are to appear underlined or not.

On the left, you can see that visited links are shown in a different color to help visitors know what they have already seen.

Often, the **:hover** and **:active** pseudo-classes (covered on the next page) are used to alter the appearance of a link when a user hovers over or clicks on it.

##### RESPONDING TO USERS

###### : hover , : active , : focus

There are three pseudo-classes that allow you to change the appearance of elements when a user is interacting with them.

**:hover**

This is applied when a user hovers over an element with a pointing device such as a mouse. This has commonly been used to change the appearance of links and buttons when a user places their cursor over them. It is worth noting that such events do not work on devices that use touch screens (such as the iPad) because the screen is not able to tell when someone is hovering their finger over an element.

**:active**

This is applied when an element is being activated by a user; for example, when a button is being pressed or a link being clicked. Sometimes this is used to make a button or link feel more like it is being pressed by changing the style or position of the element slightly.

**:focus**

This is applied when an element has focus. Any element that you can interact with, such as a link you can click on or any form control can have focus.

##### ATTRIBUTE SELECTORS

There are also a set of attribute selectors that allow you to create rules that apply to elements that have an attribute with a specific value.

|  |  |  |
| --- | --- | --- |
| SELECTOR | MEANING | EXAMPLE |
| EXISTENCE | **[]**  Matches a specific attribute | **p[class]**  Targets any **<p>** element with an |
|  | (whatever its value) | attribute called **class** |
| EqUALITY | **[=]**  Matches a specific attribute with | **p[class="dog"]**  Targets any **<p>** element with |
|  | a specific value | an attribute called **class** whose |
|  |  | value is **dog** |
| SPACE | **[~=]**  Matches a specific attribute | **p[class~="dog"]**  Targets any **<p>** element with |
|  | whose value appears in a space- | an attribute called **class** whose |
|  | separated list of words | value is a list of space-separated |
|  |  | words, one of which is **dog** |
| PREFIX | **[^=]**  Matches a specific attribute | **p[attr^"d"]**  Targets any **<p>** element with |
|  | whose value begins with a | an attribute whose value begins |
|  | specific string | with the letter "d" |
| SUBSTRING | **[\*=]**  Matches a specific attribute | **p[attr\*"do"]**  Targets any **<p>** element with an |
|  | whose value contains a specific | attribute whose value contains |
|  | substring | the letters "do" |
| SUFFIX | **[$=]**  Matches a specific attribute | **p[attr$"g"]**  Targets any **<p>** element with an |
|  | whose value ends with a specific | attribute whose value ends with |
|  | string | the letter "g" |

#### BOXES

BOX DIMENSIONS

###### Width , height

By default a box is sized just big enough to hold its contents. To set your own dimensions for a box you can use the **height** and **width** properties.

The most popular ways to specify the size of a box are to use pixels, percentages, or ems. Traditionally, pixels havebeen the most popular method because they allow designers to accurately control their size.

When you use percentages, the size of the box is relative to the size of the browser window or, if the box is encased within another box, it is a percentage of the size of the containing box.

When you use ems, the size of the box is based on the size of text within it. Designers have recently started to use percentages and ems more for measurements as they try to create designs that are flexible across devices which have different-sized screens.

LIMITING WIDTH

###### Min – width , max – width

Some page designs expand and shrink to fit the size of the user's screen. In such designs, the **min-width** property specifies the smallest size a box can be displayed at when the browser window is narrow, and the **max-width** property indicates the maximum width a box can stretch to when the browser window is wide.

These are very helpful properties to ensure that the content of pages are legible (especially on the smaller screens of handheld devices). For example, you can use the **max-width** property to ensure that lines of text do not appear too wide within a big browser window and you can use the **min-width** property to make sure that they do not appear too narrow.

LIMITING HEIGHT

###### Min – height , max – height

In the same way that you might want to limit the width of a box on a page, you may also want to limit the height of it. This is achieved using the **min-height** and **max-height** properties.

It also shows you what happens when the content of the box takes up more space than the size specified for the box.

OVERFLOWING CONTENT

###### Overflow

The **overflow** property tells the browser what to do if the content contained within a box is larger than the box itself. It can have one of two values:

**hidden**

This property simply hides any extra content that does not fit in the box.

**scroll**

This property adds a scrollbar to the box so that users can scroll to see the missing content.

The **overflow** property is particularly handy because some browsers allow users to adjust the size of the text to appear as large or as small as they want. If the text is set too large then the page can become an unreadable mess. Hiding the overflow on such boxes helps prevent items overlapping on the page.

##### BORDER, MARGIN & PADDING

Every box has three available properties that can be adjusted to control its appearance:

1

BORDER

Every box has a border (even if it is not visible or is specified to be 0 pixels wide). The border separates the edge of one box from another.

2

MARGIN

Margins sit outside the edge of the border. You can set the width of a margin to create a gap between the borders of two adjacent boxes.

3

PADDING

Padding is the space between the border of a box and any content contained within it. Adding padding can increase the readability of its contents.

##### If you specify a width for a box, then the borders, margin, and padding are added to its width and heigh.

hello

BORDER

###### Border-width

The **border-width** property is used to control the width of a border. The value of this property can either be given in pixels or using one of the following values:

**thin**

**medium**

**thick**

(You cannot use percentages with this property.)

You can control the individual size of borders using four separate properties:

**Border-top-width border-right-width border-bottom-width border-left-width**

You can also specify different widths for the four border values in one property, like so:

**border-width: 2px 1px 1px 2px;**

The values here appear in clockwise order: top, right, bottom, left.

###### Border-style

You can control the style of a border using the **border-style** property. This property can take the following values:

**Solid** a single solid line

**dotted** a series of square dots (if your border is 2px wide, then the dots are 2px squared with a 2px gap between each dot)

**dashed** a series of short lines

**double** two solid lines (the value of the **border-width** property creates the sum of the two lines)

**groove** appears to be carved into the page

**ridge** appears to stick out from the page

**inset** appears embedded into the page

**outset** looks like it is coming out of the screen

**hidden** / **none** no border is shown

You can individually change the styles of different borders using:

**border-top-style**

**border-left-style**

**border-right-style**

**border-bottom-style**

###### Border-color

You can specify the color of a border using either RGB values, hex codes or CSS color names.

It is possible to individually control the colors of the borders on different sides of a box using:

**border-top-color**

**border-right-color**

**border-bottom-color**

**border-left-color**

It is also possible to use a shorthand to control all four border colors in the one property:

**border-color:darkcyan deeppink darkcyan deeppink;**

The values here appear in clockwise order: top, right, bottom, left.

##### SHORTHAND

###### border

The **border** property allows you to specify the width, style and color of a border in one property (and the values should be coded in that specific order).

###### Padding

The **padding** property allows you to specify how much space should appear between the content of an element and its border.

The value of this property is most often specified in pixels (although it is also possible to use percentages or ems ). If a percentage is used, the padding is a percentage of the browser window (or of the containing box if it is inside another box).

Please note: If a width is specified for a box, padding is added onto the width of the box.

You can specify different values for each side of a box using:

**padding-top**

**padding-right**

**padding-bottom**

**padding-left**

Or you can use a shorthand (where the values are in clockwise order: top, right, bottom, left):

**padding: 10pxn 5pxn 3pxn 1px;**

###### Margin

The **margin** property controls the gap between boxes. Its value is commonly given in pixels, although you may also use percentages or ems.

If one box sits on top of another, margins are collapsed , which means the larger of the two margins will be used and the smaller will be disregarded.

Please note: If the width of a box is specified then the margin is added to the width of the box.

You can specify values for each side of a box using:

**margin-top**

**margin-right**

**margin-bottom**

**margin-left**

you can also use the shorthand (where the values are in clockwise order: top, right, bottom, left):

**margin: 1px 2px 3px 4px;**

Sometimes you might see the following, which means that the left and right margins should be 10 pixels and the top and bottom margins should be 20 pixels:

**margin: 10px 20px;**

This same shorthand shown above can also be applied to padding.

##### CHANGE INLINE/BLOCK

###### display

The **display** property allows you to turn an inline element into a block-level element or vice versa, and can also be used to hide an element from the page.

The values this property can take are:

**inline**

This causes a block-level element to act like an inline element.

**block**

This causes an inline element to act like a block-level element.

**inline-block**

This causes a block-level element to flow like an inline element, while retaining other features of a block-level element.

**none**

This hides an element from the page. In this case, the element acts as though it is not on the page at all (although a user could still see the content of the box if they used the ***view source*** option in their browser).

If you use this property, it is important to note that inline boxes are **not** supposed to create block-level elements.

##### HIDING BOXES

###### Visibility

The **visibility** property allows you to hide boxes from users but It leaves a space where the element would have been.

This property can take two values:

**Hidden**

This hides the element.

**Visible**

This shows the element.

If the **visibility** of an element is set to **hidden**, a blank space will appear in its place.

If you do not want a blank space to appear, then you should use the **display** property with a value of **none** instead.

###### Border-image

The **border-image** property applies an image to the border of any box. It takes a background image and slices it into nine pieces.

This property requires three pieces of information:

1: The URL of the image

2: Where to slice the image

3: What to do with the straight edges;

the possible values are:

**stretch** stretches the image

**repeat** repeats the image

**round** like repeat but if the tiles do not fit exactly, scales the tile image so they will do

The box must also have a border width for the image to be shown.

The **-moz-border-image** and **-webkit-border-image**  are the syntax for border image.

###### Box-shadow

The **box-shadow** property allows you to add a drop shadow around a box. It works just like the **text-shadow** property that you met on page 288. It must use at least the first of these two values as well as a color:

HORIZONTAL OFFSET

Negative values position the shadow to the left of the box.

VERTICAL OFFSET

Negative values position the shadow to the top of the box.

BLUR DISTANCE

If omitted, the shadow is a solid line like a border.

SPREAD OF SHADOW

If used, a positive value will cause the shadow to expand in all directions, and a negative value will make it contract.

The **inset** keyword can also be used before these values to create an inner-shadow.

the **-moz-box-shadow** and **-webkit-box-shadow** properties are the syntax of box shadow.

###### Border-radius

CSS3 introduces the ability to create rounded corners on any box, using a property called **border-radius**. The value indicates the size of the radius in pixels.

Older browsers that do not support this property will show a box with right-angled corners.

The **-moz-border-radius** and **-webkit-border-radius**  are the syntax of border radius.

You can specify individual values for each corner of a box using:

**border-top-right-radius**

**border-bottom-right-radius**

**border-bottom-left-radius**

**border-top-left-radius**

You can also use a shorthand of these four properties (in clockwise order: top, right, bottom, left). For example:

**border-radius: 5px, 10px, 5px, 10px;**

To create more complex shapes, you can specify different distances for the horizontal and the vertical parts of the rounded corners.

For example, this will create a radius that is wider than it is tall:

**border-radius: 80px 50px; 50px**



**80px**

You can target just one corner using the individual properties for that corner:

**border-top-left-radius: 80px 50px;**

There is also a shorthand for targetting all four corners at once; first you specify the four horizontal values, then the four vertical values,.

You can even create a circle by taking a square box and making the **border-radius** the same height as the square.

BULLET POINT STYLES

###### list-style-type

The **list-style-type** property allows you to control the shape or style of a bullet point (also known as a **marker**).

It can be used on rules that apply to the **<ol>**, **<ul>**, and **<li>** elements.

UNORDERED LISTS

For an unordered list you can use the following values:

**none**

**disc**

**circle**

**square**

ORDERED LISTS

For an ordered (numbered) list you can use the following values:

**decimal**

1 2 3

**decimal-leading-zero**

01 02 03

**lower-alpha**

a b c

**upper-alpha**

A B C

**lower-roman**

i. ii. iii.

**upper-roman**

I II III

##### IMAGES FOR BULLETS

###### list-style-image

You can specify an image to act as a bullet point using the **list-style-image** property.

The value starts with the letters **url** and is followed by a pair of parentheses. Inside the parentheses, the path to the image is given inside double quotes.

This property can be used on rules that apply to the **<ul>** and **<li>** elements.

##### POSITIONING THE MARKER

###### list-style-position

Lists are indented into the page by default and the **list-style- position** property indicates whether the marker should appear on the inside or the outside of the box containing the main points.

This property can take one of two values:

**outside**

The marker sits to the left of the block of text. (This is the default behaviour if this property is not used.)

**inside**

The marker sits inside the box of text (which is indented).

##### LIST SHORTHAND

###### list-style

As with several of the other CSS properties, there is a property that acts as a shorthand for list styles. It is called **list-style**, and it allows you to expressthe markers' style, image and position properties in any order.

##### TABLE PROPERTIES

Several properties will put them together in a single example

**width** to set the width of the table

**padding** to set the space between the border of each table cell and its content

**text-transform** to convert the content of the table headers to uppercase

**letter-spacing, font-size** to add additional styling to the content of the table headers

**border-top, border-bottom** to set borders above and below the table headers

**text-align** to align the writing to the left of some table cells and to the right of the others

**background-color** to change the background color of the alternating table rows

**:hover** to highlight a table row when a user's mouse goes over it

Here are some tips for styling tables to ensure they are clean and easy to follow:

GIVE CELLS PADDING

If the text in a table cell either touches a border (or another cell), it becomes much harder to read. Adding padding helps to improve readability.

DISTINGUISH HEADINGS

Putting all table headings in bold (the default style for the **<th>** element) makes them easier to read. You can also make headings uppercase and then either add a background color or an underline to clearly distinguish them from content.

SHADE ALTERNATE ROWS

Shading every other row can help users follow along the lines. Use a subtle distinction from the normal color of the rows to keep the table looking clean.

ALIGN NUMERALS

You can use the **text-align** property to align the content of any column that contains numbers to the right, so that large numbers are clearly distinguished from smaller ones.

ONLINE EXTRA

There are more examples of using CSS to style tables in the tools section of the website.

##### BORDER ON EMPTY CELLS

###### empty-cells

If you have empty cells in your table, then you can use the **empty-cells** property to specify whether or not their borders should be shown.

Since browsers treat empty cells in different ways, if you want to explicitly show or hide borders on any empty cells then you should use this property.

It can take one of three values:

**show**

This shows the borders of any empty cells.

**hide**

This hides the borders of any empty cells.

**inherit**

If you have one table nested inside another, the **inherit** value instructs the table cells to obey the rules of the containing table.

##### GAPS BETWEEN CELLS

###### border-spacing, border-collapse

The **border-spacing** property allows you to control the distance between adjacent cells. By default, browsers often leave a small gap between each table cell, so if you want to increase or decrease this space thenthe **border-spacing** property allows you to control the gap.

The value of this property is usually specified in pixels. You can specify two values if desired to specify separate numbers for horizontal and vertical spacing.

When a border has been used on table cells, where two cells meet, the width of lines would be twice that of the outside edges. It is possible to collapse adjacent borders to prevent this using the **border-collapse** property.

Possible values are:

**collapse**

Borders are collapsed into a single border where possible. (**border-spacing** will be ignored and cells pushed together, and **empty-cells** properties will be ignored.)

**separate**

Borders are detached from each other. (**border-spacing** and **empty-cells** will be obeyed.)

##### STYLING FORMS

##### STYLING TEXT INPUTS

**font-size** sets the size of the text entered by the user.

**color** sets the text color, and **background-color** sets the background color of the input.

**border** adds a border around the edge of the input box, and **border-radius** can be used to create rounded corners (for browsers that support this property).

The **:focus** pseudo-class is used to change the background color of the text input when it is being used, and the **:hover** psuedo-class applies the same styles when the user hovers over them.

**background-image** adds a background image to the box. Because there is a different image for each input, we are using an attribute selector looking for the value of the **id** attribute on each input.

STYLING SUBMIT BUTTONS

Here are some properties that can be used to style submit buttons.

**color** is used to change the color of the text on the button.

**text-shadow** can give a 3D look to the text in browsers that support this property.

**border-bottom** has been used to make the bottom border of the button slightly thicker, which gives it a more 3D feel.

**background-color** can make the submit button stand out from other items around it. (Creating a consistent style for all buttons helps users understand how they should interact with the site.

The **:hover** pseudo-class has been used to change the appearance of the button when the user hovers over it. In this case, the background changes, the text gets darker, and the thicker border is applied to the top of the button.

##### STYLING FIELDSETS & LEGENDS

Fieldsets are particularly helpful in determining the edges of a form. In a long form they can help group together related information within it.

The legend is used to indicate what information is required in the fieldset.

Properties commonly used with these two elements include:

**width** is used to control the width of the fieldset. In this example, the width of the fieldset forces the form elements to wrap onto a new line in the correct place. (If it were wider, the items might sit on one line.)

**color** is used to control the color of text.

**background-color** is used to change the color behind these items.

**border** is used to control the appearance of the border around the fieldset and/or legend.

**border-radius** is used to soften the edges of these elements in browsers that support this property.

**Padding** can be used to add space inside these elements.

##### CURSOR STYLES

###### cursor

The **cursor** property allows you to control the type of mouse cursor that should be displayed to users.

For example, on a form you might set the cursor to be a hand when the user hovers over it.

Here are the most commonly used values for this property:

**auto**

**crosshair**

**default**

**pointer**

**move**

**text**

**wait**

**help**

**url("cursor.gif");**

You should only use these values to add helpful information for users in places they would expect to see that cursor.

##### NORMAL FLOW

###### position:static

In normal flow, each block-level element sits on top of the next one. Since this is the default way in which browsers treat HTML elements, you do not need a CSS property to indicate that elements should appear in normal flow, but the syntax would be:

**position: static;**

##### RELATIVE POSITIONING

###### position:relative

Relative positioning moves an element in relation to where it would have been in normal flow.

For example, you can move it 10 pixels lower than it would have been in normal flow or 20% to the right.

You can indicate that an element should be relatively positioned using the **position** property with a value of **relative**.

You then use the offset properties (**top** or **bottom** and **left** or **right**) to indicate how far to move the element from where it would have been in normal flow.

To move the box up or down, you can use either the **top** or **bottom** properties.

To move the box horizontally, you can use either the **left** or **right** properties.

The values of the box offset properties are usually given in pixels, percentages or ems.

##### ABSOLUTE POSITIONING

###### position:absolute

When the **position** property is given a value of **absolute**, the box is taken out of normal flow and no longer affects the position of other elements on the page. (They act like it is not there.)

The box offset properties (**top** or **bottom** and **left** or **right**) specify where the element should appear in relation to its containing element.

##### FIXED POSITIONING

###### position:fixed

Fixed positioning is a type of absolute positioning that requires the **position** property to have a value of **fixed**.

It positions the element in relation to the browser window. Therefore, when a user scrolls down the page, it stays in the exact same place. It is a good idea to try this example in your browser to see the effect.

To control where the fixed position box appears in relation to the browser window, the box offset properties are used.

##### OVERLAPPING ELEMENTS

###### z-index

When you use relative, fixed, or absolute positioning, boxes can overlap. If boxes do overlap, the elements that appear later in the HTML code sit on top of those that are earlier in the page.

If you want to control which element sits on top, you can use the **z-index** property. Its value is a number, and the higher the number the closer that element is to the front. For example, an element with a **z-index** of **10** will appear over the top of one with a **z-index** of **5**.

##### FLOATING ELEMENTS

###### float

The **float** property allows you to take an element in normal flow and place it as far to the left or right of the containing element as possible.

Anything else that sits inside the containing element will flow around the element that is floated.

When you use the **float** property, you should also use the **width** property to indicate how wide the floated element should be. If you do not, results can be inconsistent but the box is likely to take up the full width of the containing element.

##### USING FLOAT TO PLACE ELEMENTS SIDE-BY-SIDE

A lot of layouts place boxes next to each other. The **float** property is commonly used to achieve this.

When elements are floated, the height of the boxes can affect where the following elements sit.

##### CLEARING FLOATS

###### clear

The **clear** property allows you to say that no element (within the same containing element) should touch the left or right- hand sides of a box. It can take the following values:

**left**

The left-hand side of the box should not touch any other elements appearing in the same containing element.

**right**

The right-hand side of the box will not touch elements appearing in the same containing element.

**both**

Neither the left nor right-hand sides of the box will touch elements appearing in the same containing element.

**none**

Elements can touch either side.

##### CREATING MULTI-COLUMN LAYOUTS WITH FLOATS

Many web pages use multiple columns in their design. This is achieved by using a **<div>** element to represent each column. The following three CSS properties are used to position the columns next to each other:

**width**

This sets the width of the columns.

**float**

This positions the columns next to each other.

**margin**

This creates a gap between the columns.

##### SCREEN SIZES

Different visitors to your site will have different sized screens that show different amounts of information, so your design needs to be able to work on a range of different sized screens.

iPhone 4 iPad 2

Size: 3.5 inches Size: 9.7 inches

Resolution: 960 x 640 pixels Resolution: 1024 x 768 pixels

Resolution refers to the number of dots a screen shows per inch. Some devices have a higher resolution than desktop computers and most operating systems allow users to adjust the resolution of their screens.

13" MacBook 27" iMac

S ize: 13.3 inches Size: 27 inches

Resolution: 1280 x 800 pixels Resolution: 2560 x 1440 pixels

It is interesting to note that the higher the resolution, the smaller the text appears. Many mobile devices have screens that are higher resolution than their desktop counterparts.

##### PAGE SIZES

Because screen sizes and display resolutions vary so much, web designers often try to create pages of around 960-1000 pixels wide (since most users will be able to see designs this wide on their screens).

##### FIXED WIDTH LAYOUTS

Fixed width layout design do not change size as the user increases or decreases the size of their browser window. Measurements tend to be given in pixels.

ADVANTAGES

* Pixel values are accurate at controlling size and positioning of elements.
* The designer has far greater control over the appearance and position of items on the page than with liquid layouts.
* You can control the lengths of lines of text regardless of the size of the user's window.
* The size of an image will always remain the same relative to the rest of the page.

DIS-ADVANTAGES

* You can end up with big gaps around the edge of a page.
* If the user's screen is a much higher resolution than the designer's screen, the page can look smaller and text can be harder to read.
* If a user increases font sizes, text might not fit into the allotted spaces.
* The design works best on devices that have a site or resolution similar to that of desktop or laptop computers.
* The page will often take up more vertical space than a liquid layout with the same content.

##### LIQUID LAYOUTS

Liquid layout designs stretch and contract as the user increases or decreases the size of their browser window. They tend to use percentages.

ADVANTAGES

* + Pages expand to fill the entire browser window so there are no spaces around the page on a large screen.
  + If the user has a small window, the page can contract to fit it without the user having to scroll to the side.
  + The design is tolerant of users setting font sizes larger than the designer intended (because the page can stretch).

DIS-ADVANTAGES

* If you do not control the width of sections of the page then the design can look very different than you intended, with unexpected gaps around certain elements or items squashed together.
* If the user has a wide window, lines of text can become very long, which makes them harder to read.
* If the user has a very narrow window, words may be squashed and you can end up with few words on each line.
* If a fixed width item (such as an image) is in a box that is too small to hold it (because the user has made the window smaller) the image can overflow over the text.

##### A FIXED WIDTH LAYOUT

To create a fixed width layout, the width of the main boxes on a page will usually be specified in pixels (and sometimes their height, too).

The fixed width layout will stay the same width no matter what size the browser window is, whereas the liquid layout will stretch (or shrink) to fill the screen.

##### A LIQUID LAYOUT

The liquid layout uses percentages to specify the width of each box so that the design will stretch to fit the size of the screen.

##### LAYOUT GRIDS

Composition in any visual art (such as design, painting, or photography) is the placement or arrangement of visual elements — how they are organized on a page. Many designers use a grid structure to help them position items on a page, and the same is true for web designers.

Grids set consistent proportions and spaces between items which helps to create a professional looking design.

While a grid might seem like a restriction, in actual fact it:

* Creates a continuity between different pages which may use different designs
* Helps users predict where to find information on various pages
* Makes it easier to add new content to the site in a consistent way
* Helps people collaborate on the design of a site in a consistent way

##### POSSIBLE LAYOUTS: 960 PIXEL WIDE

12 COLUMN GRID

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | **94** | **0** | **px** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | **46** | **0** | **px** |  |  |  |  |  |  |  |  | **46** | **0** | **px** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **30** | **0** | **px** |  |  |  |  | **30** | **0** | **px** |  |  |  |  | **30** | **0** | **px** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | **2** | **20 p** | **x** |  |  | **2** | **20 p** | **x** |  |  | **2** | **20 p** | **x** |  |  | **2** | **20 p** | **x** |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **14** | **0** | **px** | **14** | **0** | **px** | **14** | **0** | **px** | **14** | **0** | **px** | **14** | **0** | **px** | **14** | **0** | **px** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

These two pages illustrate a 960 pixel wide, 12 column grid. They demonstrate how it is possible to create a wide range of column layouts using this one grid.

The page is 960 pixels wide and there are 12 equal sized columns (shown in gray), each of which is is 60 pixels wide.

Each column has a margin set to 10 pixels, which creates a a gap of 20 pixels between each column and 10 pixels to the left and right-hand sides of the page.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **3** | **0** | **0 px** |  |  |  |  |  |  |  |  | **62** | **0** | **px** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | **2** | **20 p** | **x** |  |  |  |  |  |  |  |  | **7** | **00 p** | **x** |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | **46** | **0** | **px** |  |  |  |  |  | **2** | **20 p** | **x** |  |  | **2** | **20 p** | **x** |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **30** | **0** | **px** |  |  |  |  | **30** | **0** | **px** |  |  | **14** | **0** | **px** | **14** | **0** | **px** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | **2** | **20 p** | **x** |  |  | **2** | **20 p** | **x** |  | **14** | **0** | **px** | **14** | **0** | **px** | **14** | **0** | **px** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

##### CSS FRAMEWORKS

CSS frameworks aim to make your life easier by providing the code for common tasks, such as creating layout grids, styling forms, creating printer-friendly versions of pages and so on. You can include the CSS framework code in your projects rather than writing the CSS from scratch.

ADVANTAGES

* + They save you from repeatedly writing code for the same tasks.
  + They will have been tested across different browser versions (which helps avoid browser bugs).

DISADVANTAGES

They often require that you use class names in your HTML code that only control the presentation of the page (rather than describe its content).

In order to satisfy a wide variety of needs, they often contain more code than you need for your particular web page (commonly referred to as code “bloat”).

INTRODUCING THE 960.GS CSS FRAMEWORK

One of the most popular uses of CSS frameworks is in creating grids to layout pages. There are several grid frameworks out there, but the one we will be looking at over the next few pages is the 960 Grid System (available at [**www.960.gs**](http://www.960.gs)).

960.gs provides a style sheet that you can include in your HTML pages. Once our page links to this style sheet, you can provide the appropriate classes to your HTML code and it will create multiple column layouts for you. The 960.gs website also provides templates you can download to help design your pages using a 12 column grid. (In addition, there is a variation on the grid that uses 16 columns.)

To create a 12 column grid, an element that contains the entire page is given a **class** attibute whose value is **container\_12**. This sets the content of the page to be 960 pixels wide and indicates that we are using a 12 column grid.

There are different classes for blocks that take up 1, 2, 3, 4, and up to 12 columns of the grid. Each block uses class names such as **grid\_3** (for a block that stretches over three columns), **grid\_4** (for a block that stretches over 4 columns) and and so on through to **grid\_12** (for a box that is the full width of the page). These columns all float to the left, and there is a 10 pixel margin to the left and the right of each one.

There are several other grid- based CSS frameworks available online, such as those at:

**blueprintcss.org lessframework.com developer.yahoo.com/yui/ grids/**

##### USING THE 960.GS GRID

Below you can see a sample layout of a page just like the fixed width page example. On the next page, we will recreate this using the 960.gs stylesheet. Instead of writing our own CSS to control layout, we will need to add classes to the HTML indicating how wide each section should be.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | **94** | **0** | **px** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | **94** | **0** | **px** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | **94** | **0** | **px** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **30** | **0** | **px** |  |  |  |  | **30** | **0** | **px** |  |  |  |  | **30** | **0** | **px** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | **94** | **0** | **px** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

The 960.gs style sheet has taken care of the layout, creating the correct width for the columns and setting the spaces between them. Therefore, the only rules we needed to add are shown on this page. These rules:

* Control the font and the position of the text in the boxes
* Set the background colors for the boxes
* Set the height of the feature and article boxes
* Add a margin to the top and bottom of each box

##### MULTIPLE STYLE SHEETS

###### @import

Some web page authors split up their CSS style rules into separate style sheets. For example, they might use one style sheet to control the layout and another to control fonts, colors and so on

Some authors take an even more **modular** approach to stylesheets, creating separate stylesheets to control typography, layout, forms, tables, even different styles for each sub-section of a site

There are two ways to add multiple style sheets to a page:

1: Your HTML page can link to one style sheet and that stylesheet can use the **@import** rule to import other style sheets.

2: In the HTML you can use a separate **<link>** element for each style sheet.

This stylesheet then uses the **@import** rule to import the **typography.css** and **tables.css** files.

#### IMAGES

#### CONTROLLING SIZES OF IMAGES IN CSS

You can control the size of an image using the **width** and **height** properties in CSS, just like you can for any other box.

Specifying image sizes helps pages to load more smoothly because the HTML and CSS code will often load before the images, and telling the browser how much space to leave for an image allows it to render the rest of the page without waiting for the image to download.

You might think that your site is likely to have images of all different sizes, but a lot of sites use the same sized image across many of their pages.

For example, an e-commerce site tends to show product photos at the same size. Or, if your site is designed on a grid, then you might have a selection of image sizes that are commonly used on all pages, such as:

Small portrait: 220 x 360

Small landscape: 330 x 210

Feature photo: 620 x 400

Whenever you use consistently sized images across a site, you can use CSS to control the dimensions of the images, instead of putting the dimensions in the HTML.

First you need to determine the sizes of images that will be used commonly throughout the site, then give each size a name.

For example:

**Small**

**Medium**

**large**

Where the **<img>** elements appear in the HTML, rather than using **width** and **height** attributes you can use these names as values for the **class** attribute.

##### ALIGNING IMAGES USING CSS

Rather than using the **<img>** element's **align** attribute, web page authors are increasingly using the **float** property to align images. There are two ways that this is commonly achieved:

1: The **float** property is added to the class that was created to represent the size of the image (such as the **small** class in our example).

2: New classes are created with names such as **align-left** or **align-right** to align the images to the left or right of the page.

These class names are used in addition to classes that indicate the size of the image.

It is also common to add a margin to the image to ensure that the text does not touch their edges.

##### CENTERING IMAGES USING CSS

By default, images are inline elements. This means that they flow within the surrounding text. In order to center an image, it should be turned into a block- level element using the **display** property with a value of **block**.

Once it has been made into a block-level element, there are two common ways in which you can horizontally center an image:

1: On the containing element, you can use the **text-align** property with a value of **center**.

2: On the image itself, you can use the use the **margin** property and set the values of the left and right margins to **auto**.

You can specify class names that allow any element to be centered, in the same way that you can for the dimensions or alignment of images.

##### BACKGROUND IMAGES

###### background-image

The **background-image** property allows you to place an image behind any HTML element. This could be the entire page or just part of the page. By default, a background image will repeat to fill the entire box.

The path to the image follows the letters **url**, and it is put inside parentheses and quotes.

##### REPEATING IMAGES

###### background-repeat

###### background-attachment

The **background-repeat** property can have four values:

**repeat**

The background image is repeated both horizontally and vertically (the default way it is shown if the **background- repeat** property isn't used).

**repeat-x**

The image is repeated horizontally only (as shown in the first example on the left).

**repeat-y**

The image is repeated vertically only.

**no-repeat**

The image is only shown once.

The **background-attachment** property specifies whether a background image should stay in one position or move as the user scrolls up and down the page. It can have one of two values:

**fixed**

The background image stays in the same position on the page.

**scroll**

The background image moves up and down as the user scrolls up and down the page.

##### BACKGROUND POSITION

###### background-position

When an image is not being repeated, you can use the **background-position** property to specify where in the browser window the background image should be placed.

This property usually has a pair of values. The first represents the horizontal position and the second represents the vertical.

**left top**

**left center**

**left bottom**

**center top**

**center center**

**center bottom**

**right top**

**right center**

**right bottom**

If you only specify one value, the second value will default to **center**.

You can also use a pair of pixels or percentages. These represent the distance from the top left corner of the browser window (or containing box). The top left corner is equal to **0% 0%**. The example shown, with values of **50% 50%,** centers the image horizontally and vertically.

##### SHORTHAND

###### background

The **background** property acts like a shorthand for all of the other background properties you have just seen, and also the **background-color** property.

The properties must be specified in the following order, but you can miss any value if you do not want to specify it.

1: **background-color**

2: **background-image**

3: **background-repeat**

4: **background-attachment**

5: **background-position**

CSS3 will also support the use of multiple background images by repeating the **background** shorthand. Because few browsers supported this property at the time of writing, it was not commonly used.

IMAGE ROLLOVERS & SPRITES

Using CSS, it is possible to create a link or button that changes to a second style when a user moves their mouse over it (known as a **rollover**) and a third style when they click on it.

This is achieved by setting a background image for the link or button that has three different styles of the same button (but only allows enough space to show one of them at a time).

You can see the image we are using in this example on the right. It actually features two buttons on the one image.

When the user moves their mouse over the element, or clicks on it, the position of the background image is moved to show the relevant image.

When a single image is used for several different parts of an interface, it is known as a **sprite**.

You can add the logo and other interface elements, as well as buttons to the image.

The advantage of using sprites is that the web browser only needs to request one image rather than many images, which can make the web page load faster.

The **background-position** property is used to move the image in order to show the button in the right state.

When the user hovers over a link, the **:hover** pseudo-class has a rule that moves the **background-position** of the image to show a different state for that button.

Similarly, when the user clicks on the link, the **:active** pseudo- class has a rule to show the third state for that button.

##### CSS3: GRADIENTS

###### background-image

CSS3 is going to introduce the ability to specify a gradient for the background of a box. The gradient is created using the **background-image** property and, at the time of writing, different browsers required a different syntax.

Since it is not supported by all browsers, it is possible to specify a background image for the box first (which would represent the gradient) and then provide the CSS alternatives for browsers that support gradients.

##### CONTRAST OF BACKGROUND IMAGES

If you want to overlay text on a background image, the image must be low contrast in order for the text to be legible.

HIGH CONTRAST

The majority of photographs have quite a high contrast, which means that they are not ideal for use as a background image.

LOW CONTRAST

Image editing applications such as Photoshop and GIMP have tools that allow you to manually adjust your images to have lower contrast.

SCREEN

To overlay text on an image with high contrast, you can place a semi-transparent background color (or "screen") behind the text to improve legibility.

#### HTML5 LAYOUT

##### NEW HTML5 LAYOUT ELEMENTS

HTML5 introduces a new set of elements that allow you to divide up the parts of a page. The names of these elements indicate the kind of content you will find in them. They are still subject to change, but that has not stopped many web page authors using them already.

**<body>**

**<div id="page">**

**<footer>**

**<aside>**

**<div id="content">**

**<article>**

**<article>**

**<header>**

**<nav>**

However, many of the **<div>** elements have been replaced by new HTML5 layout elements.

For example, the header sits inside a new **<header>** element, the navigation in a **<nav>** element, and the articles are in individual **<article>** elements.

The point of creating these new elements is so that web page authors can use them to help describe the structure of the page. For example, screen reader software might allow users to ignore headers and footers and get straight to the content. Similarly, search engines might place more weight on the content in an **<article>** element than that in the **<header>** or **<footer>** elements. I think you will agree that it also makes the code easier to follow.

##### HEADERS & FOOTERS

###### <header> <footer>

The **<header>** and **<footer>** elements can be used for:

* The main header or footer that appears at the top or bottom of every page on the site.
* A header or footer for an individual **<article>** or **<section>** within the page.

The **<footer>** element contains copyright information, along with links to the privacy policy and terms and conditions.

Each individual **<article>** and **<section>** element can also have its own **<header>** and **<footer>** elements to hold the header or footer information for that section within the page.

##### NAVIGATION

###### <nav>

The **<nav>** element is used to contain the major navigational blocks on the site such as the primary site navigation.

##### ARTICLES

###### <article>

The **<article>** element acts as a container for any section of a page that could stand alone and potentially be syndicated.

This could be an individual article or blog entry, a comment or forum post, or any other independent piece of content.

If a page contains several articles (or even summaries of several articles), then each individual article would live inside its own

**<article>** element.

The **<article>** elements can even be nested inside each other.

##### ASIDES

###### <aside>

The **<aside>** element has two purposes, depending on whether it is inside an **<article>** element or not.

When the **<aside>** element is used inside an **<article>** element, it should contain information that is related to the article but not essential to its overall meaning. For example, a pullquote or glossary might be considered as an aside to the article it relates to.

When the **<aside>** element is used outside of an **<article>** element, it acts as a container for content that is related to the entire page. For example, it might contain links to other sections of the site, a list of recent posts, a search box, or recent tweets by the author.

##### SECTIONS

###### <section>

The **<section>** element groups related content together, and typically each section would have its own heading.

For example, on a homepage there may be several **<section>** elements to contain different sections of the page, such as latest news, top products, and newsletter signup.

Because the **<section>** element groups related items together, it may contain several distinct **<article>** elements that have a common theme or purpose.

Alternatively, if you have a page with a long article, the **<section>** element can be used to split the article up into separate sections.

The **<section>** element should not be used as a wrapper for the entire page (unless the page only contains one distinct piece of content). If you want a containing element for the entire page, that job is still best left to the **<div>** element.

##### HEADING GROUPS

###### <hgroup>

The purpose of the **<hgroup>** element is to group together a set of one or more **<h1>** through **<h6>** elements so that they are treated as one single heading.

For example, the **<hgroup>** element could be used to contain both a title inside an **<h2>** element and a subtitle within an **<h3>** element.

##### FIGURES

###### <figure> <figcaption>

It can be used to contain any content that is referenced from the main flow of an article (not just images).

It is important to note that the article should still make sense if the content of the **<figure>** element were moved (to another part of the page, or even to a different page altogether).

For this reason, it should only be used when the content simply references the element (and not for something that is absolutely integral to the flow of a page).

Examples of usage include:

* Images
* Videos
* Graphs
* Diagrams
* Code samples
* Text that supports the main body of an article

The **<figure>** element should also contain a **<figcaption>** element which provides a text decription for the content of the **<figure>** element. In this example, you can see a **<figure>** has been added inside the **<article>** element.

##### SECTIONING ELEMENTS

###### <div>

It may seem strange to follow these new elements by revisiting the **<div>** element again. (After all, the new elements are often going to be used in its place.)

However, the **<div>** element will remain an important way to group together related elements, because you should not be using these new elements that you have just met for purposes other than those explicitly stated.

Where there is no suitable element to group a set of elements, the **<div>** element will still be used. In this example, it is used as a wrapper for the entire page.

##### LINKING AROUND BLOCK-LEVEL ELEMENTS

HTML5 allows web page authors to place an **<a>** element around a block level element that contains child elements. This allows you to turn an entire block into a link.

This is not a new element in HTML5, but it was not seen as a correct usage of the **<a>** element in earlier versions of HTML.

##### CSS PROPERTIES

background-attachment property

background-color

background-image property

background-image property(gradients)

background-position property

background property

background-repeat property

border-bottom-color

border-bottom-style

border-collapse property

border-color

border-image property

border-left-color

border-left-style

border property

border-radius property

border-right-color

border-right-style

border-spacing property

border-style

border-top-color

border-top-style

bottom property

box-shadow property

clear property

color

cursor property

display property

empty-cells property

float property

float property (images)

@font-face

font-face

font-family

font-size

font-style property

height (images)

hsl / hsla

left property

letter-spacing

line-height

list-style-image property

list-style-position property

list-style property

list-style-type property

margin-bottom property

margin-left property

margin property

margin-right property

margin-top property

opacity

padding-bottom property

padding-left property

padding property 3

padding property (tables)

padding-right property

padding-top property

position property

rgba

right property

text-align

text-decoration

text-indent

text-shadow

text-transform

top property

vertical-align

visibility property

width (images)

width property (floating elements)

width property (tables)

word-spacing

z-index property

PSEUDO-CLASSES, ELEMENTS & RULES

:active

:first-letter

:first-line s

:focus

:hover

highlighting table rows

:link

:visited

@font-face

@import rule

!important