Thursday: CSS

CSS

The key to understanding how CSS works is imagine that there is an invisible box around every HTML element.

CSS allows you to create rules that control the way that each individual box (and the contents of that box) is presented.

How do you implement these rules?

1. By using inline styling
2. By using the style tag
3. By using an external CSS file

Why use external CSS?

1. Allows all pages to use the same style rules.
2. Keeps the content separate from how the page looks.
3. Means you can change the styles used across all pages by altering just one file.

What do you use to change these rules?

Selectors - used to select the content you want to style.

Selectors

Types of selectors

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>, <h3> elements

Class Selector	Matches an element whose class attributes has a value that matches the one specified after the period (or full stop) symbol	.note {} 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 id attribute has a value that matches the one specified after the pound or hash symbol	#introduction {} Targets the element whose id attribute has a value of introduction
Child Selector	Matches an element that is a direct child of another	li>a {} Targets any <a> elements that are children of an element (but no other <a> elements in the page)
Descendant Selector	Matches an element that is a descendant of another specified element (not just a direct child of that element)	p a {} Targets any <a> elements that sit inside a <p> element, even if there are other elements nested between them
Adjacent Sibling Selector	Matches an element that is the next sibling of another	h1+p {} Targets the first <p> element after any <h1> element (but not other <p> elements)
General Sibling Selector	Matches an element that is a sibling of another, although it does not have to be the directly preceding element	h1~p {} If you had two <p> elements that are siblings of an <h1> element, this rule would apply to both

How do these rules work with each other?

- Last rule - if the two selectors are identical, the latter of the two will take precedence.
- Specificity - if one selector is more specific than the others, the more specific rule will take precedence over the more general one.
- 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.

Understanding how CSS rules cascade means you can write simpler style sheets because you can create generic rules that apply to most elements and then override the properties on individual elements that need to appear differently.

Text

Properties that allow you to control the appearance of text can be split into 2 groups:

1. Those that directly affect the font and its appearance (including the typeface, whether it is regular bold or italic and the size of the text)
2. Those that would have the same effect on text no matter what font you were using (including the colour of text and the spacing between words and letters)

Typeface terminology

Serif

Serif fonts have extra details on the ends of the main strokes of the letters (serifs)

Sans-serif

Have straight ends to letters, and therefore have a much cleaner design.

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

Monospace

Fixed width (same width) for every letter.

Commonly used for code because they align nicely, making the text easier to follow.

Specifying typefaces: font-family

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 font.

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

Size of type: font-size

Ways to specify the font-size:

1. Pixels - commonly used because they allow web designers very precise control over how much space their text takes up.
2. Percentages - the font size is defined as a percentage of the default text size. E.g. default size of text in browsers is 16px hence 75% = 12px and 200% = 32px
3. Ems - a unit equal to the computed font-size for the element to which the em is applied. When em units are declared on child elements that don't have a font-size defined, they will inherit their font-size from their parent, or from another ancestor element, possible going all the way back to the root element on the document.

More Font Choice: @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.

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

Leading: line-height

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 opt 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 line-height is equivalent to the leading.

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

A good starter setting is around 1.4em to 1.5em

Letter and Word Spacing: letter-spacing, word-spacing

Kerning is 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.

When specifying a value for these properties, it should be given in ems and it will be added on top of the default value for the font

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 , although it does have this effect when used with table cells (<td> and <th> elements)

It is more commonly used with inline elements such as <img>, , or elements. When used with these elements, it performs a task very similar to HTML align attribute used on the <img> element.

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. Property can take negative values to push text off the browser window.

Pseudo-elements

These are keywords added to a selector that lets you style a specific part of the selected element. They act as visible elements on a web page that aren't "in the DOM" or created from HTML but are instead inserted directly from CSS. This allows you to work on the styling without cluttering the markup.

Styling links: :link, :visited

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

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:

1. :hover - this is applied when a user hovers over an element with a pointing device such as a mouse
2. :active - this is applied when an element is being activated by a user (pressing or clicking a button)
3. :focus - applied when an element has focus e.g. form. Focus occurs when a browser discovers that you are ready to interact with an element on the page. E.g. when your cursor is in a form input ready to accept typing, that element is said to have focus.

When pseudo-classes are used, they should appear in this order:

- :link

- :visited
- :hover
- :focus
- :active

Attribute selectors

Selector	Meaning	Example
Existence	[] Matches a specific attribute (whatever its value)	p[class] Targets any <p> element with an attribute called class
Equality	[=] Matches a specific attribute with a specific value	p[class=”dog”] Targets any <p> element with an attribute called class whose value is dog
Space	[~=] Matches a specific attribute whose value appears in a space-separated list of words	p[class~=”dog”] Targets any <p> element with an attribute called class whose value is a list of space-separated words, one of which is dog
Prefix	[^=] Matches a specific attribute whose value begins with a specific string	p[attr^=”d”] Targets any <p> element with an attribute whose value begins with the letter “d”
Substring	[*=] Matches a specific attribute whose value contains a specific substring	p[attr*=”do”] Targets any <p> element with an attribute whose value contains the letters “do”
Suffix	[$=] Matches a specific attribute whose value ends with a specific string	p[attr$=”g”] Targets any <p> element with an attribute whose value ends with the letter “g”

Boxes

To set your own dimensions for a box you can use the height and width properties.

Traditionally, pixels have been 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 the 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.

min-width specifies the smallest size a box can be displayed at when the browser window is narrow.

max-width indicates the maximum width a box can stretch to when the browser window is wide.

Similarly, you can limit the height of a box using the min-height and max-height properties.

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:

1. **hidden** - this property simply hides any extra content that does not fit in the box.
2. **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**- separates the edge of one box from another
2. **margin** - creates a gap between the borders of two adjacent boxes
3. **padding** - the space between the border of a box and any content contained within it. Adding padding increases the readability of content

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.

The margin property is not inherited by child elements

Centering content

If you want to center a box on a page (or centre it inside the element that it sits in) you can set the left-margin and the right-margin to auto.

In order to centre a box on the page, you need to set a width for the box (otherwise it will take up the full width of the page)

Once you've specified the width of the box, setting the left and right margins to auto will make the browser put an equal gap on each side of the box.

The text-align property is inherited by child elements. You therefore also need to specify the text-align property on the centered box if you do not want the text inside to be centered.

Change inline/block: display

This 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.

Hiding boxes: visibility

Allows you to hide boxes from users but it leaves a space where the element would have been.

This property can take two values:

1. hidden - this hides the element
2. visible - this shows the element

Rounded corners: border-radius

This property allows you to create rounded corners on any box, using a property called border-radius. The value indicates the size of the radius in pixels.

A note on vendor prefixes

CSS vendor prefixes are a way for browser makers to add support for new CSS features before those features are fully supported in all browsers. This may be done during a sort of testing and

experimentation period where the browser manufacturer is determining exactly how these new CSS features will be implemented.

Common prefixes are:

- Android, Chrome, iOS & Safari: -webkit-
- Firefox: -moz-
- Internet Explorer: -ms-
- Opera: -o-

In most cases, to use a brand new CSS style property, you take the standard CSS property and add the prefix for each browser. The prefixed versions would always come first (in any order you prefer) while the normal CSS property will come last.

Ex:

```
p{
border-radius: 10px;
-moz-border-radius: 10px;
-webkit-border-radius: 10px;
}
```

Some browsers have a different syntax for certain properties than others do, so don't assume that the browser-prefixed version of a property is exactly the same as the standard property.

For example, to create a CSS gradient, you use the linear-gradient property. Firefox, Opera and modern versions of Chrome and Safari use that property with the appropriate prefix while early versions of Chrome and Safari use the prefixed property -webkit-gradient

The reason that you always end your declaration with the normal, non-prefixed version of the CSS property is so that when a browser does support the rule, it will use that one. Remember how CSS is read. The later rules take precedence over earlier ones if the specificity is the same, so a browser would read the vendor version of a rule and use that if it does not support the normal one, but once it does, it will override the vendor version with the actual CSS rule.

Box-shadows: box-shadow

Allows you to add a drop shadow around a box. It must use at least the first of these two values as well as colour:

1. Horizontal offset- negative values position the shadow to the left of the box.
2. Vertical offset - negative values position the shadow to the top of the box
3. Blur distance - if omitted, the shadow is a solid line like a border

4. Spread of shadow - if used, a positive will cause the shadow to expand in all directions and a negative value will make it contract.

The onset keyword can also be used before these values to create an inner shadow

List, Tables and Forms

Bullet point styles: list-style-type

Allows you to control the shape or style of a bullet point (marker)

Values for unordered lists

- None
- Disc
- Circle
- square

Values for ordered lists

- Decimal
- Decimal leading zero
- Lower-alpha
- Upper-alpha
- Lower-roman
- Upper-roman

Positioning the marker: list-style-position

Property indicates whether the marker should appear on the inside or outside of the box containing the main points.

1. outside - marker sits to the left of the block of text (this is the default behaviour)
2. inside - marker sits inside the box of text (which is indented)

Shorthand

List-style: marker, image, position;

Table Properties

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

- Give cells padding

- Distinguish heading
- Shade alternate rows
- Align numerals
- Border on empty cells

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 heading

Putting all table headings in bold makes them easier to read. You can also make headings uppercase then either add a background colour or an underline to clearly distinguish them from content.

Shade alternate rows

Shading every other row can help other users follow along the lines. Use a subtle distinction from the normal colour 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 the smaller ones.

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.

Possible values are:

- 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, then the 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.)

Layout

Controlling the position of the elements: position

CSS has the following positioning schemes that allow you to control the layout of the page:

- normal flow
- relative positioning
- absolute positioning
- Fixed positioning

Normal flow: static

Every block-level element appears on a new line, causing each item to appear lower down the page than the previous one. (default behaviour)

Relative positioning: relative

This moves an element from the position it would be in normal flow, shifting it to the top, right, bottom, or left of where it would have been placed. This does not affect the position of surrounding elements; they stay in the position they would be in normal flow.

Absolute positioning: absolute

This positions the element in relation to its containing element. It is taken out of the normal flow, meaning that it does not affect the position of any surrounding elements (as they simply ignore the space it would have taken up).

Absolutely positioned elements move as users scroll up and down the page.

Fixed positioning: fixed

This is a form of absolute positioning that positions the element in relation to the browser window, as opposed to the containing element.

Elements with fixed positioning do not affect the position of surrounding elements and they do not move when the user scrolls up or down the page.

Floating elements: float

Floating an element allows you to take that element out of normal flow and position it to the far left or right of a containing box. The floated element becomes a block-level element around which other

content can flow.

When using the float property, you should also use the width property to indicate how wide the floated element should be.

When elements are floated, the height of the boxes can affect where the following elements sit. The clear property is used to correct this.

clear

Allows you to say that no element (within the same containing element) should touch the left or right-hand sides of the box.

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 site 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. E.g. an element with a z-index of 10 will appear over the top of the heading as the user scrolls down the page.

Referred also to as stacking context (as if the blocks have been stacked on top of each other on a z-axis)

Parents of floated elements

Problem

If a containing element only contains floated elements, some browsers will treat is as zero pixels tall.

Solution

Add two CSS rules to the containing element.

- Overflow property is given a value auto
- Width property is set to 100%

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

The z-index property is used to control which elements sit on top (the higher the number, the closer it is to the top)

Creating multi-column layouts with floats

This is achieved using a element to represent each column.

The CSS properties used are:

1. width- sets the width of the columns
2. float - this positions the columns next to each other
3. margin - this creates a gap between columns

Fixed width layouts

Fixed width layout designs 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 not 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.

Disadvantages

- You can end up with big gaps around the edge of the page
- If the user's screen is a much higher resolution than the designer's screen, the page could look smaller and text can be harder to reach.
- If a user increases font sizes, text might not fit in the allocated 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

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 the side.
- The design is tolerant of users setting font sizes larger than the designer intended (because the page can stretch)

Disadvantages

- 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 a 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.

CSS Practice

Use this for some CSS practice: **http://flukeout.github.io/ (https://slack-redir.net/link?url=http%3A%2F%2Fflukeout.github.io%2F)**