Block-level Elements (HTML)

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Examples of block-level elements:

- (*generic)
- <h1> - <h6>
- <p>
- <form>

Inline Elements (HTML)

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline element inside a paragraph.

Examples of inline elements:

- (*generic)
- <a>
- <img>

*Generic indicates that the browser does not add preset margin, padding, or coloration to these elements

Syntax:

<opening tag> *content* </closing tag>

<self-closing-tag> (*pre HTML5* = <self-closing-tag/>)

'Classing' Elements (CSS)

Adding a class to an element allows you to define common, standard styling that can be applied to multiple elements.
You determine the class names, therefore it is important that you name them in a way which quickly indicates it's purpose. There is a lot of debate surrounding this topic - I recommend reading:

https://smacss.com/book/categorizing
http://nicolasgallagher.com/about-html-semantics-front-end-architecture/

We will come back to this with CSS...

HTML5 defines layout specific elements

These elements are semantically (relating to meaning in language or logic) named in order to make the markup more intuitive for developers and browsers. While the name of these elements give them meaning, they do not come with default styling such as margin, padding, or font colors.

header	Defines a header for a document or a section
nav	Defines a container for navigation links
section	Defines a section in a document
article	Defines an independent self-contained article
aside	Defines content aside from the content (like a sidebar)
footer	Defines a footer for a document or a section
details	Defines additional details
summary	Defines a heading for the details element

CSS

CSS stands for **C**ascading **S**tyle **S**heets
Styles 'Cascade' downward, being inherited until a more specific style is declared.

Styling can be added to HTML elements in 3 ways:

- Inline - using a **style attribute** in HTML elements
- Internal - using a **<style> element** in the HTML <head> section
- External - using one or more **external CSS files**

As of today, External CSS should be the only way your sites are developed. However, creating HTML emails requires inline-css.

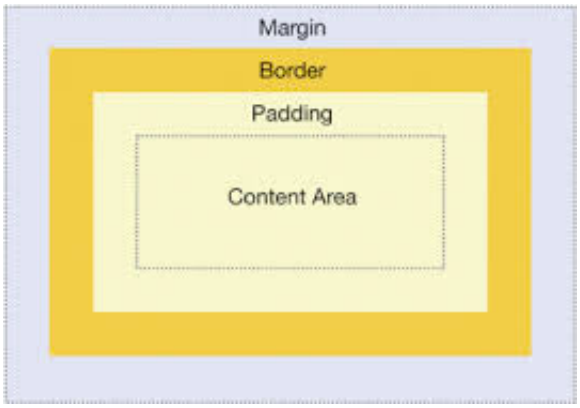
Syntax:

```
element {
	attribute: value;
	attribute: value;
}

.class { // can be used over and over on one page
	attribute: value;
	attribute: value;
}

#id { // can only be used once per page
	attribute: value;
	attribute: value;
}
```

CSS Box Model

Every HTML element is essentially a box.

The CSS **border** property defines a visible border around an HTML element.
The CSS **padding** property defines a padding (space) inside the border.
The CSS **margin** property defines a margin (space) outside the border.



Margin is unique in that it doesn't affect the size of the box itself per se, but it affects *other* content interacting with the box, and thus an important part of the CSS box model.

The size of the box itself is calculated like this:

Width	width + padding-left + padding-right + border-left + border-right
Height	height + padding-top + padding-bottom + border-top + border-bottom

You can eliminate this effect by placing the following attribute on any element: **box-sizing: border-box;**
This will result in the element maintaining it's set width or height even when padding or border is declared.

CSS Positioning

The **`position`** property specifies the type of positioning method used for an element (static, relative, fixed or absolute)

position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

z-index

Z-Index can be used to overlap elements.

The z-index property specifies the stack order of an element, placing it in front of, or behind, other elements.
The element with the highest z-index number (stack number) will appear above an element with a lower stack number.

```
img {
    position: absolute;
    left: 0px;
    top: 0px;
    z-index: -1;
}
```

Properties of positioning: bottom, top, left, right, overflow, z-index.

CSS Float

The Float property allows an element to step out of it's natural stacking order. They can be floated left or right - never center. Floats have been widely used in creating layouts, but there are other ways to get this effect.

Inline

It has been possible for a long time to create a grid of boxes that fills the browser width and wraps nicely (when the browser is resized), by using the `float` property.

However, the **`inline-block`** value of the `display` property makes this even easier.

inline-block elements are like inline elements but they can have a width and a height.

Flexbox!

https://css-tricks.com/snippets/css/a-guide-to-flexbox/