**Zeldman 10 CSS Layout**

**Block** box (block-level elements e.g. <p> <h2> <div>

vs.

**Inline** box (inline elements e.g. <img /> <span> these sit on a line, will wrap, don't add a return.

All boxes follow the basic model:

**Content**

**Padding**

**Border**

**Margin**

[As Zeldman says, the margin is artificially filled; actually only the border takes a color. The element's box ends with the border as shown; the margin is the space between it and other elements. Vertically stacked boxes will collapse their margins]

Margin

There's no property like this
content: 300px;

but width: 300px;

does what you'd think content: 300px does.

The default Box Model adds content (width) + padding + borders. So

#sidebar {

 width: 400px;

 padding: 50px;

 border: 2px;

}

has a width of 2+50+400+50+2 = 504px

However, you can now override how boxes are calculated. Most developers use

 [box-sizing: border-box](https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing);

which overrides the default of :content-box, so that padding and border don’t make the width: increase (clear?)

**Default is all 4 sides.**

padding: 50px

(Shorthands are important for download (what if there were hundreds of these rules) and also to show that you know your standards-based CSS. [font: is another common one, as are hex codes #abc; What are some others?]

**For 2 sides at a time use**

padding: 20px 30px;

which means

padding-top: 20px;

padding-bottom: 20px;

padding-right: 30px;

padding-left: 30px;

[For 3

padding 20px 30px 25px;

padding-top: 20px;

padding-right: 30px;

padding-left: 30px;

padding-bottom: 25px;

**Multiple values** are read clockwise [Note: no commas. Generally multiple values are separated by commas as with font-family: or rgb(). Also no apce between numbers and unit 20px; NOT 20 px;]:

padding: 5px 10px 15px 20px;

is the same as

padding-top: 5px

padding-right: 10px;

padding-bottom: 15px;

padding-left: 20px;

[Some people say remember the order is TRBL or you're in trouble.]

So padding: 20px 30px; is the same as padding: 20px 30px 20px 30px;

Example **Layout #1**. Think semantics first. *Then* design. Two <p> [but maybe picture goes in a <div> since it's NOT a paragraph?]

Then style

body {

font: normal 16px/24px Georgia, Times, serif;

}

Note what that saves in terms of markup with HTML <font> tags. [this is shorthand for

font-weight: normal;

font-size: 16px;

line-height: 24px;

font-family: Georgia, Times, serif;

]

p {

text-indent: .1em;

}

This is an example of a property that HTML didn't have. Also it applies to all <p> at once.

Then add a class for first p [or <div>

.figure {

padding: 10px;

margin-right: 1em;

float: left;

}

<div class="figure">

[this way you wouldn't need to reset text-indent: 0 because div {} has not been styled]

Make the <img /> block level

.figure img {

 display: block;

 margin-bottom: .5 em;

}

Floats: Non-floated neighbors "float" to the top and text wraps around, although floated element sits on top of non-floated box. [Also it's usually best practice to declare a width: with floats which he doesn't do here. Let's keep an eye on that]

**Layout #2**

Sketch it out:

.info

 <h1> 844px

124px

124px

 <h2> and also .content 844px

988px

.main

700px

.meta

Here we need content <div>s. He uses

<div id="page">

 <div class="entry">

 <div class="content">

 <div class="main">

 <div class="figure">

 <p> etc.

 </div> <!-- end .figure-->

 </div> <!--end main-->

 <div class="meta">

 </div>

 </div>

 </div>

 <div id="footer">

 </div>

</div>

Use of <h1> <h2> <h3 class="info"> lots of <p>

Add class names to earlier markup where needed, e.g.

.figure img {} becomes

.entry .main .figure img {}

#page {

 width: 988px;

 margin: 0 auto 40px;

}

If three values, it's Top Left/Right Bottom.

[This is a common auto margins centering technique. Note page div is a container (often use <div id="container">. Be warned that this doesn't always work.]

First element is an <h1> that he does not float (everything else is floated)

h1 {

 margin-left: 144px;

 width: 844px;

}

Float the <h3 class="info"> to the left (left floats go first then float <h2> to the right; should also work if you float it to the left:

float: left

float: left

Float other items :left or :right. Note they do have width: generally a best practice. Note (p. 210) floats *weren't designed to do layout* [any more than tables were.] Note <h1> is still not floated. [Alternative is position: absolute or :relative]

These floats will look like this. Note that they are all inside the <div class=”entry”>:

.entry h2 {

 float: right;

 width: 844px;

}

.entry .info {

 float: left;

 width: 124px;

}

.entry .content {

 float: right;

 width: 844px;

}

.entry .main {

 float: left;

 width: 700px;

}

.entry .meta {

 float: right;

 width: 124px;

}

**[On p. 211 near the bottom he says "meta" occupies the first 700px. He means "main"]**

#footer { clear: both;}

also a common technique [clear: left; and :right are sort of counter-intuitive]