The Presentation Layer (Cascading Style Sheets)

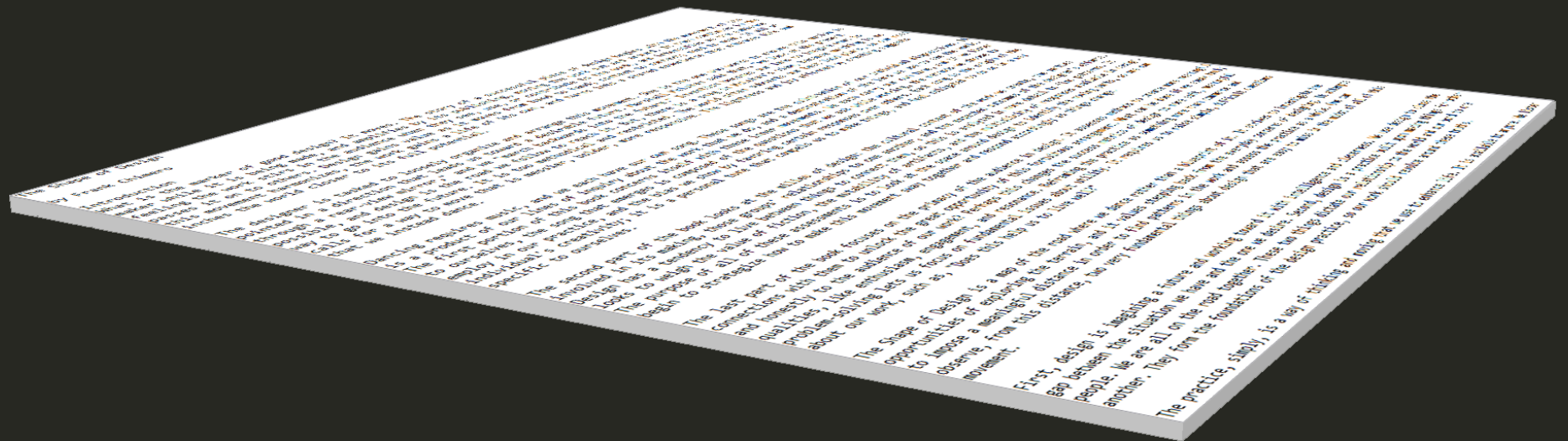
Design for web content

A conceptual model for all webpages…

The Web Standards Model

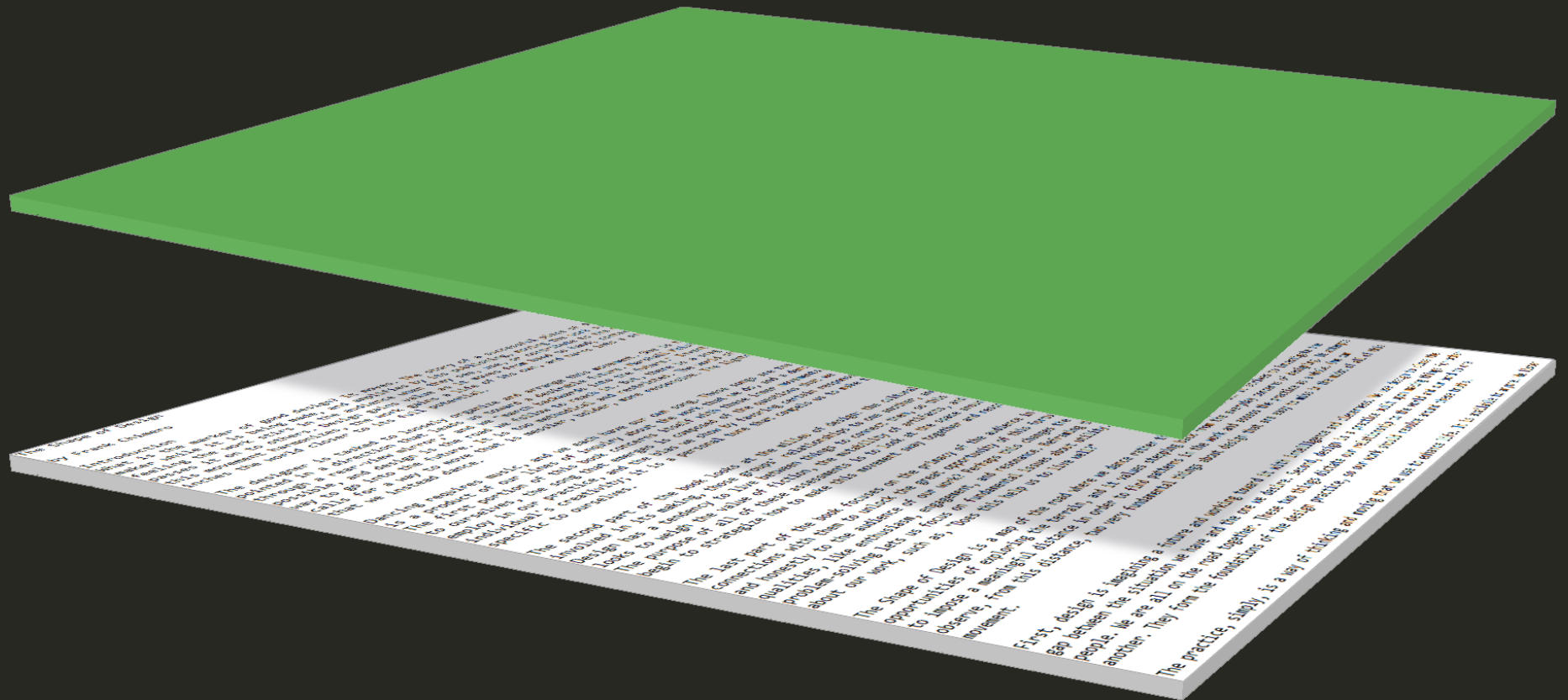
Web Standards are an important part of how the Web is regulated (technologically).
The W3C (World Wide Web Consortium) coordinates most aspects of web technologies

https://www.w3.org/



Content

Text, images etc.

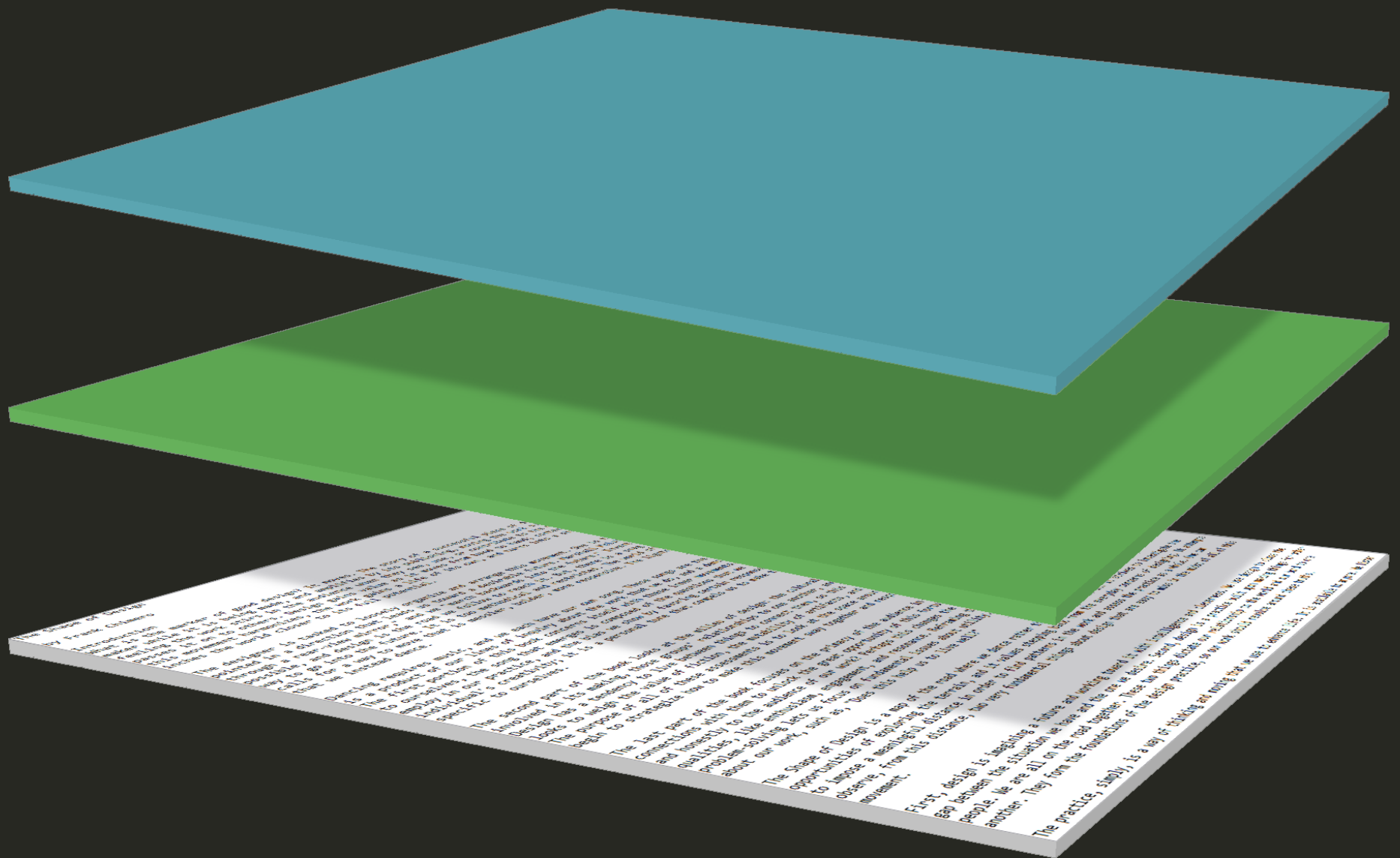


Structure

HTML

Content

Text, images etc.



Presentation

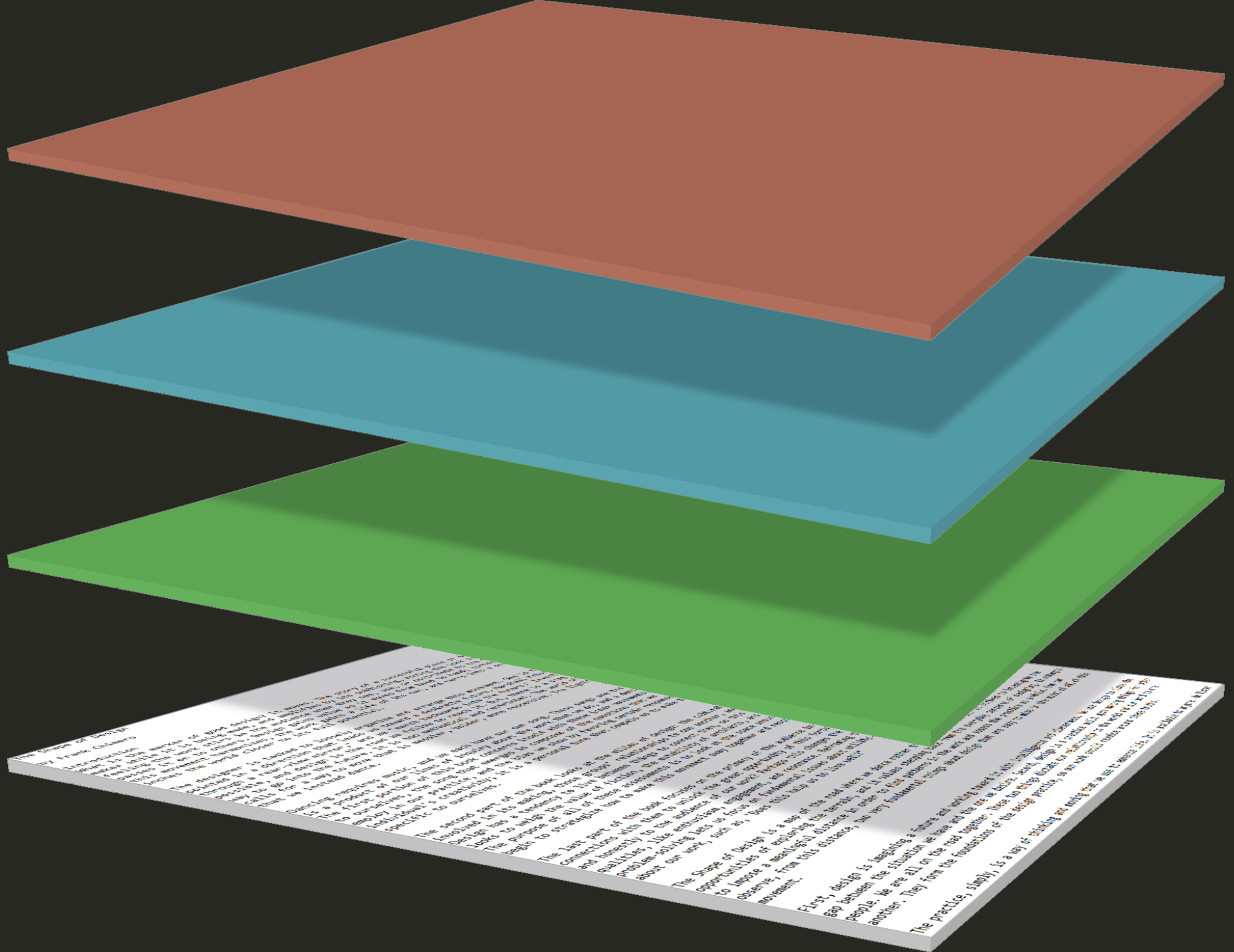
CSS

Structure

HTML

Content

Text, images etc.



Behaviour

JavaScript

Presentation

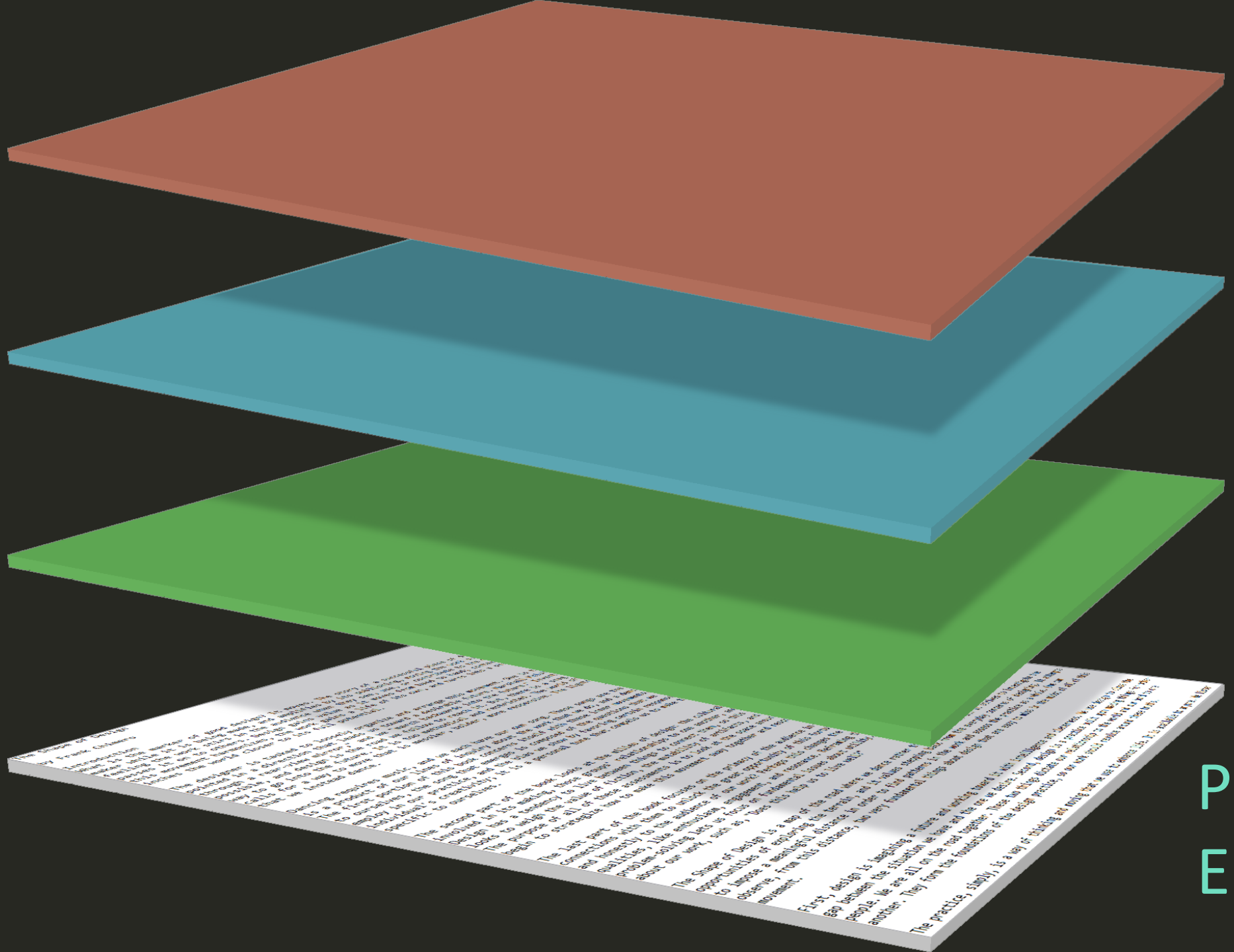
CSS

Structure

HTML

Content

Text, images etc.



Richness of experience

Progressive Enhancement

Web Standards Model in action…

structure presentation behaviour

webpage in browser

The browser loads the HTML file and then any linked resources such as the CSS file, the JavaScript file and any media files (e.g. images). A single webpage is built from many separate files.

The purpose of HTML is to provide…
Structure to documents
Meaning to content

```
<h1>The Shape of Design</h1>

<h2>Introduction</h2>

<p>What is the marker of good design? It moves. The story of a successful piece of design begins with the movement of its maker while it is being made, and amplifies by its publishing, moving the work out and around. It then continues in the feeling the work stirs in the audience when they see, use, or contribute to the work, and intensifies as the audience passes it on to others. Design gains value as it moves from hand to hand; context to context; need to need. If all of this movement harmonizes, the work gains a life of its own, and turns into a shared experience that enhances life and inches the world closer to its full potential.</p>
```

It does this by *marking-up* content with *tags*
This is **important** for non-human users

The purpose of CSS is to provide…
Visual style to documents. We refer to this as *presentation*.

```
h1 {
	font-size: 4.0em;
}
h2 {
	font-size: 2.4em;
	margin-top: 1.8em;
}
h1, h2 {
	font-family: "Playfair Display", Georgia, "Times New Roman", Times, serif;
	color: #194a82;
}
p {
	font-size: 1.2em;
	line-height: 1.4em;
	margin-top: 0.4em;
}
p + p {
	margin-top: 1.6em;
}
```

It does this using *rules* that describe what each HTML *element* should look like

The purpose of JavaScript is to add…
Behaviours to the user interface

```
var newLink = document.createElement( "a" );
var allParagraphs = document.getElementsByTagName( "p" );
var moreParagraph = allParagraphs[ 1 ];
newLink.setAttribute( "href", "#" );
newLink.setAttribute( "class", "more-link" );
newLink.innerHTML = "Read more";
moreParagraph.appendChild( newLink );
```

It does this with scripts that change how the content looks
or behaves in the browser depending on user actions

What is CSS?

- Originated in 1994 (5 years after HTML)
- CSS 1 recommended by WC3 in December 1996
- Separating content from presentation
- Smaller files = quicker load
- Much greater control over formatting
- Easier site maintenance
- Improves accessibility

https://www.w3.org/Style/CSS20/history.html

https://developer.mozilla.org/en-US/docs/Web/CSS

How does it work?

Markup (HTML) *index.html*

+

Style Rules (CSS) *style.css*

=

Rendered result in browser

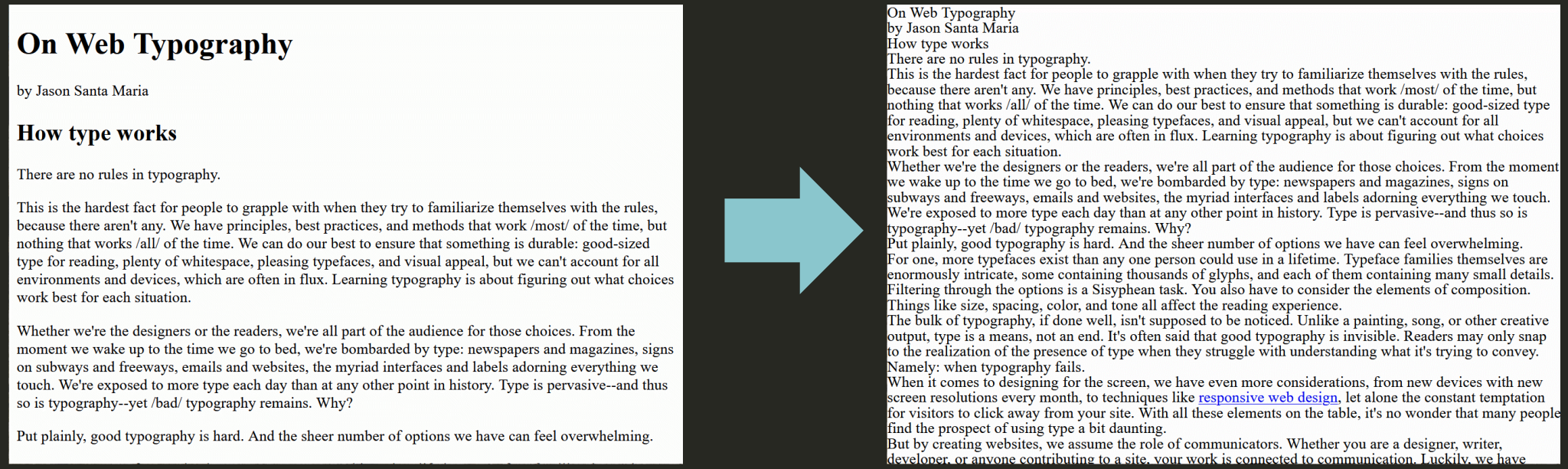
How do we add CSS to our HTML?

We don’t add it, we link to it using a <link> element.

```
<head>
    <meta charset="utf-8">
    <title>My Webpage</title>
    <link rel="stylesheet" href="style.css">
</head>
```

Linking to our CSS file allows us to maintain the separation of structure and presentation, keeping the principle of the web standards model intact. The <link> element has two mandatory attributes. The *rel* attribute defines the relationship of the linked file and the *href* attribute tells the browser where to find it.

Before adding our own styles, we first need to *reset* the browser default styling. This is good practice and helps us take full control of styling.



On Web Typography

by Jason Santa Maria

How type works

There are no rules in typography.

This is the hardest fact for people to grapple with when they try to familiarize themselves with the rules, because there aren't any. We have principles, best practices, and methods that work /most/ of the time, but nothing that works /all/ of the time. We can do our best to ensure that something is durable: good-sized type for reading, plenty of whitespace, pleasing typefaces, and visual appeal, but we can't account for all environments and devices, which are often in flux. Learning typography is about figuring out what choices work best for each situation.

Whether we're the designers or the readers, we're all part of the audience for those choices. From the moment we wake up to the time we go to bed, we're bombarded by type: newspapers and magazines, signs on subways and freeways, emails and websites, the myriad interfaces and labels adorning everything we touch. We're exposed to more type each day than at any other point in history. Type is pervasive--and thus so is typography--yet /bad/ typography remains. Why?

Put plainly, good typography is hard. And the sheer number of options we have can feel overwhelming.

On Web Typography
by Jason Santa Maria
How type works
There are no rules in typography.
This is the hardest fact for people to grapple with when they try to familiarize themselves with the rules, because there aren't any. We have principles, best practices, and methods that work /most/ of the time, but nothing that works /all/ of the time. We can do our best to ensure that something is durable: good-sized type for reading, plenty of whitespace, pleasing typefaces, and visual appeal, but we can't account for all environments and devices, which are often in flux. Learning typography is about figuring out what choices work best for each situation.
Whether we're the designers or the readers, we're all part of the audience for those choices. From the moment we wake up to the time we go to bed, we're bombarded by type: newspapers and magazines, signs on subways and freeways, emails and websites, the myriad interfaces and labels adorning everything we touch. We're exposed to more type each day than at any other point in history. Type is pervasive--and thus so is typography--yet /bad/ typography remains. Why?
Put plainly, good typography is hard. And the sheer number of options we have can feel overwhelming.
For one, more typefaces exist than any one person could use in a lifetime. Typeface families themselves are enormously intricate, some containing thousands of glyphs, and each of them containing many small details. Filtering through the options is a Sisyphean task. You also have to consider the elements of composition. Things like size, spacing, color, and tone all affect the reading experience.
The bulk of typography, if done well, isn't supposed to be noticed. Unlike a painting, song, or other creative output, type is a means, not an end. It's often said that good typography is invisible. Readers may only snap to the realization of the presence of type when they struggle with understanding what it's trying to convey. Namely: when typography fails.
When it comes to designing for the screen, we have even more considerations, from new devices with new screen resolutions every month, to techniques like responsive web design, let alone the constant temptation for visitors to click away from your site. With all these elements on the table, it's no wonder that many people find the prospect of using type a bit daunting.
But by creating websites, we assume the role of communicators. Whether you are a designer, writer, developer, or anyone contributing to a site, your work is connected to communication. Luckily, we have

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

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

Eric Meyer's Reset CSS

The most popular reset in use today is Eric Meyer's Reset CSS. It is designed to reset only those elements that will benefit from resetting and leaves others alone.

At this stage, you don’t need to know how it works. You only need to know how to use it.

Eric Meyer's Reset CSS

https://cssreset.com/scripts/eric-meyer-reset-css/

How do I apply a reset?

There are other popular reset methods, such as the HTML5 Doctor CSS Reset (an evolution of the Eric Meyer reset) but whichever method you choose, you should add the reset to the **top** of your main CSS file (style.css), before your own rules and link to it from the head of your HTML.

In HTML:

```
<link rel="stylesheet" href="style.css">
```

Note that we don't need a *type* attribute in HTML5 (it's implicit), and we don't need to close the link element either (no trailing slash).

HTML5 Doctor CSS Reset

Demo: We'll reset the browser default styles using the Meyer reset.

CSS Style Rules

How do we create our own style rules?

Style Rules

A CSS style rule is composed of a *Selector* and one or more *Declarations*. Each declaration has a *Property* and *Value* pair. Declarations are always enclosed in curly braces. The **colon** character is used as a separator between the property and value.

This rule says that all h1 elements will have the colour red (note the spelling of this property!).

Multiple Declarations

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

One CSS rule can contain many declarations.

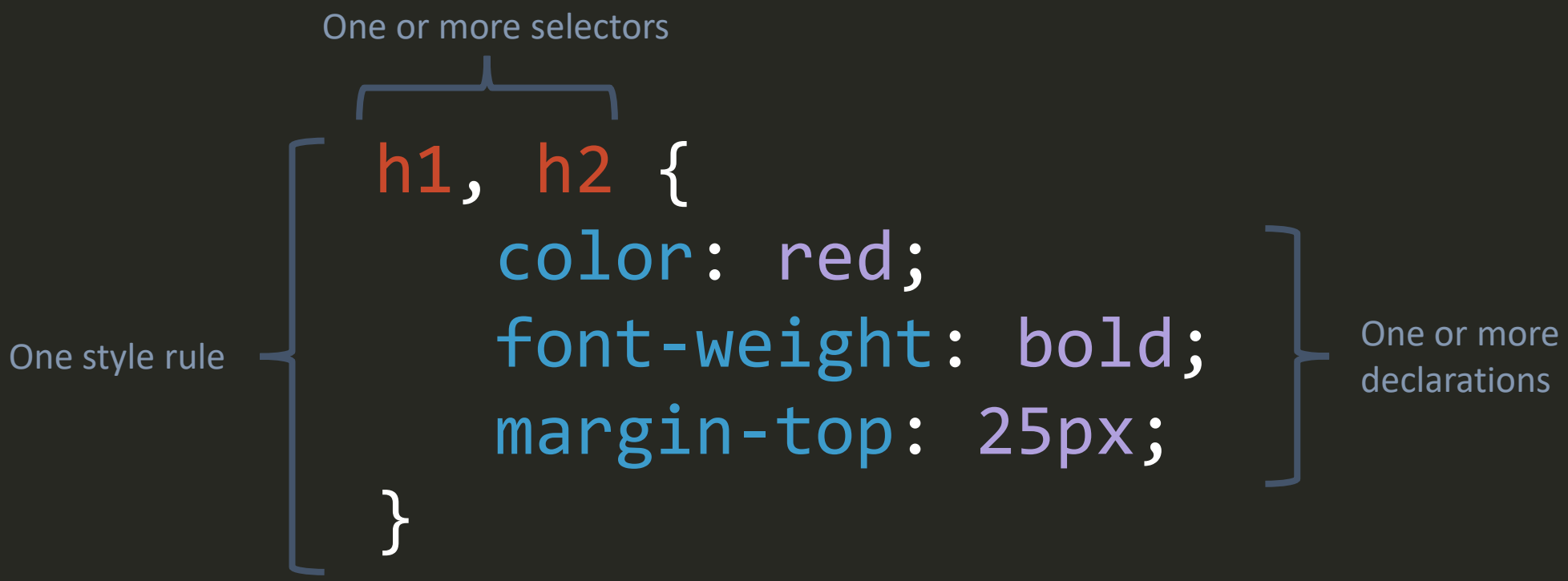
The **semi-colon** character is used as a separator between property/value pairs (declarations) within a single rule. The rule above has 3 declarations. For clarity, declarations are indented.

Grouping Selectors

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

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

Anatomy of a style rule



All style rules follow the same basic pattern (known as CSS Syntax). They begin with one or more selectors (comma separated), followed by one or more declarations (semi-colon separated) enclosed is curly brackets (braces). These elements are arranged as in the example above for clarity.

The type selector

Markup (index.html)

```
<h1>Heading</h1>
```

CSS Rule (style.css)

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

Rendered result in browser

Heading

Result: All text marked up as a first order heading <h1> will be rendered in red.

There are many selector types but the most common (and simplest) is the type selector (also known as the element selector). This selector will match any element of the specified name.

Demo: We'll add some of our own rules using the type selector.

CSS Type Properties

How do we change the default font?

System or “Web Safe” fonts

Arial
Verdana
Trebuchet
Calibri

Sans-serif fonts

Times New Roman
Georgia
Courier New
Cambria

Serif fonts

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

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

*Unless the font is being served from a web font service such as Google Fonts.

The font-family property

```
h1{
      font-family: Georgia, "Times New Roman", Times, serif;
      font-size: 3.0em;
      color: #534741;
      margin-bottom: 5px;
}
```

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

The font-size property

```
h1{
      font-family: Georgia, "Times New Roman", Times, serif;
      font-size: 3.0em;
      color: #534741;
      margin-bottom: 5px;
}
```

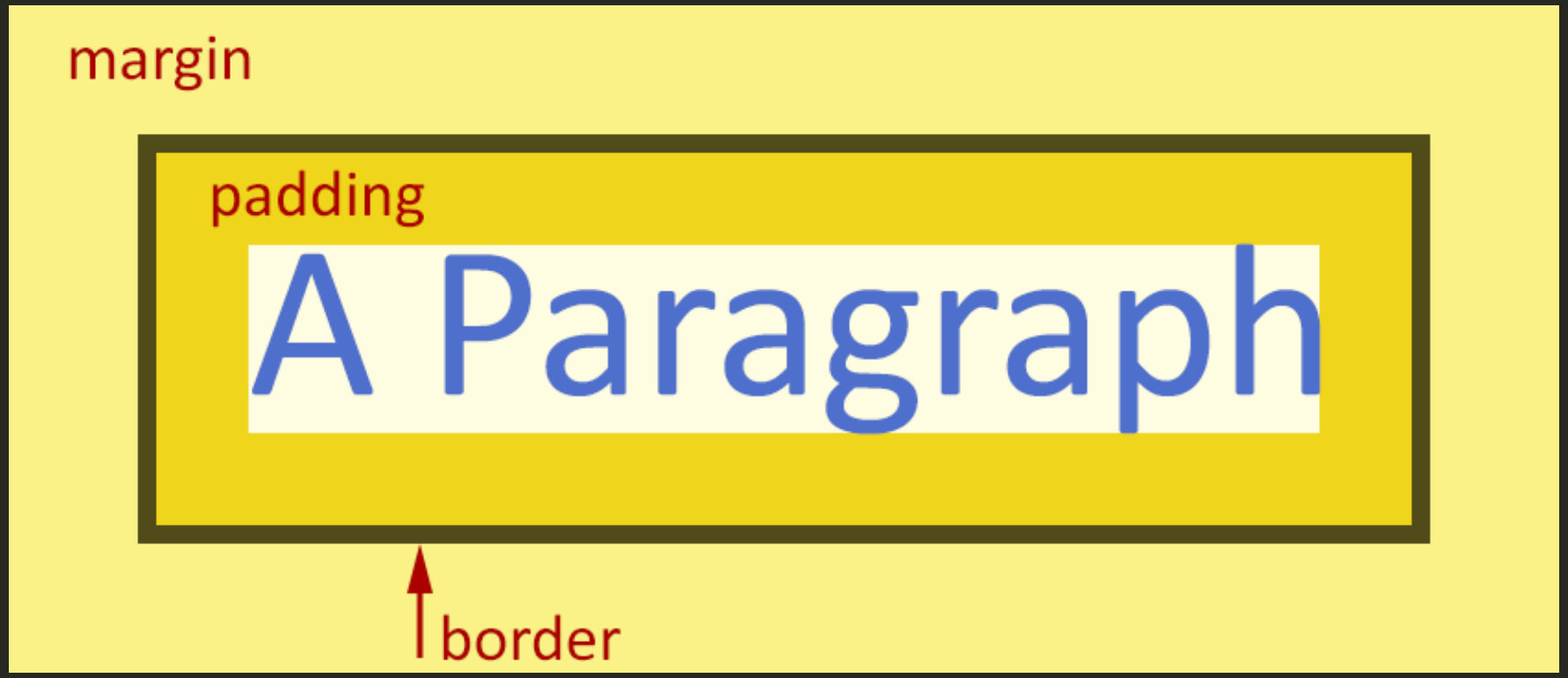
The font-size property is used to set text at the height specified. You might think that font sizes would be set using pixels (px), and you can do that if you want but, best practice is to use a proportional unit. Here, we're using the "em". By default, one em is equal to 16px but the beauty (and sometimes frustration) is that the default can change, depending on context.

Demo: We'll specify a font stack using the font-family property.

The CSS Box Model

How do we create space between elements?

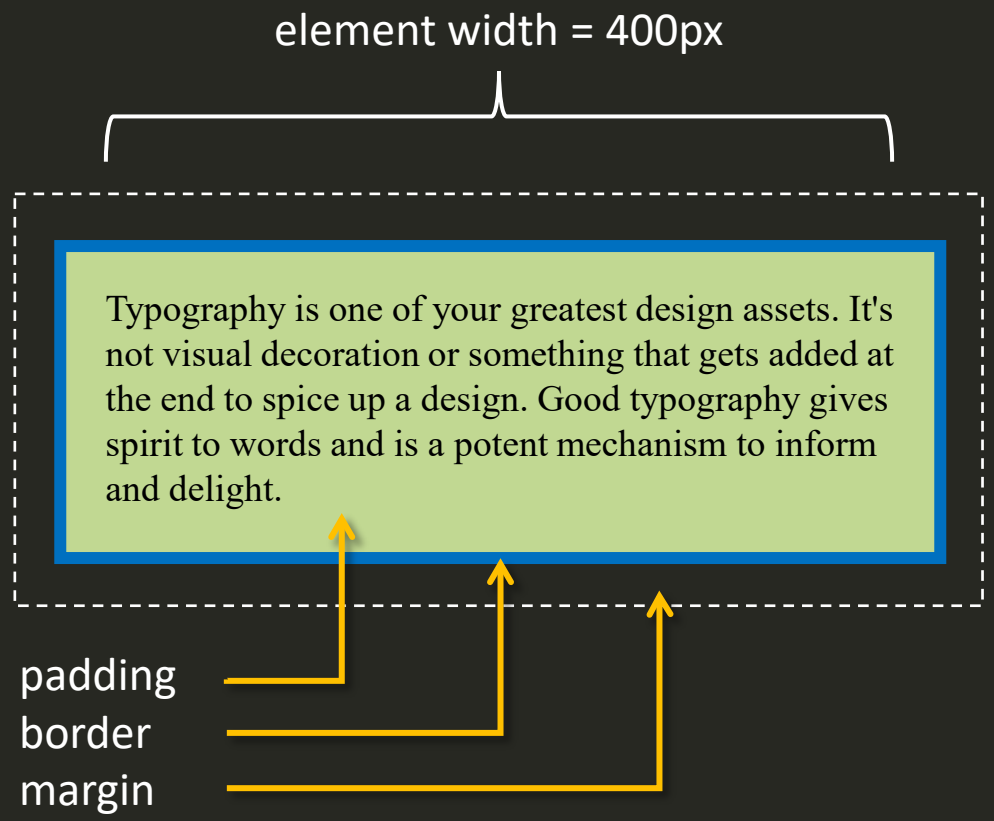
The CSS “Box Model”



The box model applies to every HTML element (block-level *and* inline), so you can think of every paragraph, heading or image as a box.

Box Model sizing

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



By default, element width does **not** include padding, border or margin, so in this case, the total width is 400 + 30 + 10 + 40 = 480px. This default method for calculating element width is often problematic because simply changing the margin on an element could break your page layout.

Note that padding takes the element background colour, the border has an independent colour, and the margin is transparent.

Box-sizing in CSS

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

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

```
* {
        box-sizing: border-box;
}
```

This sounds like a perfect solution and the only problem is that this property will not be recognised by older browsers, but current support is good. In the example above, we’re using the *universal selector* (*) to set all elements to obey this box sizing rule.

https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing

Margin and Padding

```
main {
    margin: 20px;
}
```

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

```
main {
    margin: 20px 20px 20px 20px;
}
```

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

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

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

Border

```
main {
    border: 1px solid #FFFFFF;
}
```

```
main {
    border-top: 1px solid #FFFFFF;
    border-bottom: 2px solid #EEEEEE;
}
```

The border property is slightly different because it takes three values, *width*, *style* and *colour*.

Demo: We’ll add some margins and borders.

Colour with CSS

How do we change the colour of elements?

How many colours are there?

Adobe Color Web App

Specifying colours in CSS

Colours can be specified in many ways, here are the most common:

Colour name	`color: red;`
Hexadecimal (Base 16 RGB)	`color: #FF0000;`
Hex shorthand	`color: #F00;`
RGB (Red, Green, Blue)	`color: rgb(255,0,0);`
RGBA (RGB plus Alpha)	`color: rgba(255, 0, 0, 1);`

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

Applying colours

```
body {
    color: #EEEEEE;
}
```

The color property is used to specify the foreground colour i.e. the colour of text. This property is inherited, so if applied to the body element, it will apply to all text.

```
body {
    color: #EEEEEE;
    background-color: #272822;
}
```

The background-color property is used to specify the background colour of the element to which it is applied. If we use background-color om the body element, the whole page background will display in the chosen colour.

Demo: We'll specify our colours using hexadecimal.

CSS Inheritance

Do children inherit the styles of their parents?

CSS Inheritance

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

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

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

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

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

Generally, inheritance works as you might expect.

https://www.sitepoint.com/css3-inheritance-tips-tricks/

Working with Inheritance

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

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

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

https://www.sitepoint.com/css3-inheritance-tips-tricks/

Commenting your CSS

Commenting your code is a very important habit to get into when you are learning but it also helps to make your code more readable for you and for others. The CSS commenting syntax is similar to many other coding languages (but not HTML).

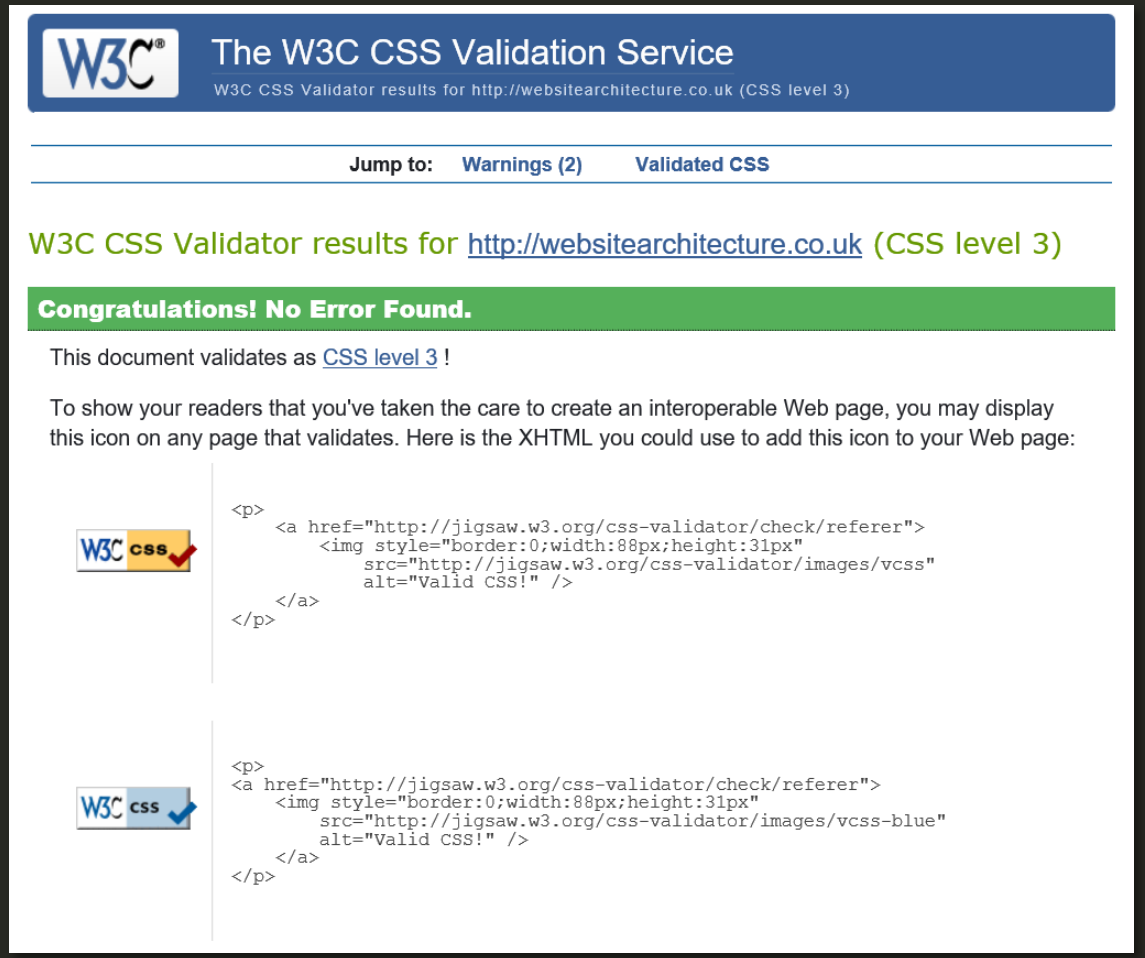
```
/* I am a single-line comment ~ useful for section headings */

/* I am a multi-line comment and you
could use me to make notes to remind
yourself why you used a particular
CSS property */
```

CSS comments begin with a slash and a star character (/*) and end with a star and a slash character (*/). Comments can be single line or multi-line. Using comments effectively is good practice.

https://developer.mozilla.org/en-US/docs/Web/CSS/Comments

Validate



The W3C CSS Validation Service

W3C CSS Validator results for http://websitearchitecture.co.uk (CSS level 3)

Jump to: Warnings (2) Validated CSS

W3C CSS Validator results for http://websitearchitecture.co.uk (CSS level 3)

Congratulations! No Error Found.

This document validates as CSS level 3 !

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

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

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

CSS can be validated in the same way as HTML using the W3C CSS Validation Service.

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

The End