HTML & CSS for Beginners

Section 2: CSS

2.1 Inline CSS

CSS stands for Cascading Style Sheets and are used to decorate and add style to HTML files to make the website/webapp to have a professional aesthetic.

As the name suggests, inline CSS means that the style is added inside of the HTML tag element. The style attribute is added to the element and its value is then set to a CSS styling using the CSS syntax.

By default the text colour is black and the font size is 16px. We can change the default style using CSS. Below is an example of inline CSS changing the text colour to green and the font size to 30px.

```
<p style="color: green; font-size: 30px;">Deciding what not to do is as important as deciding what to do.</p>
```

Deciding what not to do is as important as deciding what to do.

Note: The green square box is displayed using the VS Code's Color Picker extension. This would typically not be displayed in the text editor without the extension added.

Important Disclaimer: Inline CSS is not recommended as best practice. However, it is important to understand that this is a possibility for styling a HTML document.

2.2 Internal & External CSS

An internal CSS is created by using the <style></style> element tags nested inside of the <head></head> element tags. This allows the HTML file to use the internal CSS style for rendering the web page.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Internal CSS</title>
    <style>
        p {
            color: green;
            font-size: 30px;
        }
    </style>
```

We would use the CSS syntax inside of the style element tags. This will result in the same style behaviour as seen in the inline CSS example.

This style will target all paragraph elements.

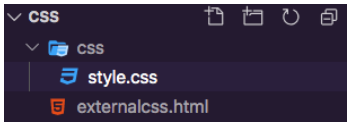
Important Disclaimer: Internal CSS is not recommended as best practice. However, it is important to understand that this is also a possibility for styling a HTML document.

To create a External CSS we would need to create a new directory within our root project directory called css. This will contain our external .css file typically named style.css.

In this .css file we would use the CSS syntax by targeting the element and then in the curly brackets defining the style/decoration for that element. We do not require the <style></style> HTML element tags in the .css file.

The HTML file can import this external css file and extract the style/decorations to apply to the HTML elements in the document.

The <link rel="stylesheet" href="css/style.css"> element tags which is a self closing element tag is used inside of the <head></head> element tags to import external CSS files.
The rel attribute describes the relationship of the file imported and the href attribute points to the location of the .css file relative to the HTML file importing it.




```css
p {
    color: green;
    font-size: 30px;
}
```

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>External CSS</title>
</head>
```

Important Disclaimer: External CSS is recommended as best practice. This allows you to break up the code into separate files which makes it much easier to update and maintain the code.

2.3 Classes & IDs

There are two attribute available to use on all HTML element tags: class and id. These attributes allow us to target our CSS style to individual elements are across multiple elements.

Two elements cannot have the same id and therefore id attributes values must be unique. The class tribute value can be shared across multiple elements.

The CSS syntax to target a element by it's id is to use the pound sign (#) followed by the id name. To target a class the syntax is to use the period sign (.) followed by the class name.

```html
    <style>
        #decide {
            color: green;
            font-size: 30px;
        }
        .blue {
            color: blue;
        }
    </style>
</head>
<body>
    <p id="decide">Deciding what not to do is as important as deciding what to do.</p>
    <p class="blue">Second paragraph.</p>
    <p class="blue">Third paragraph.</p>
</body>
```

Deciding what not to do is as important as deciding what to do.

Second paragraph.

Third paragraph.

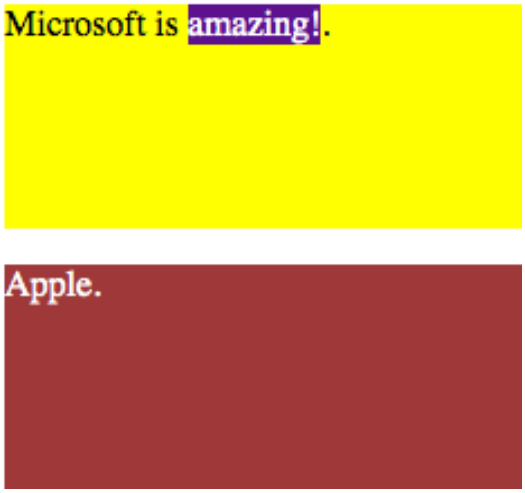
2.4 Div & Span

A element tag is a HTML section tag which groups a block elements. The division element is then used to style these elements using CSS. The division element is a block element.

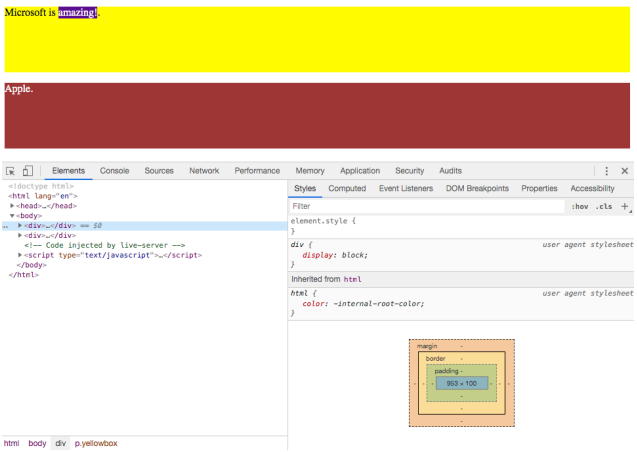
A element tag is an inline element which is used to group inline elements and style them using CSS.

There are multiple ways of selecting a colour in CSS. We can either select a colour by its name, by its hexadecimal value, RGB value and RGBA value.

```
    <style>
        .yellowbox {
            background-color: yellow;
            height: 100px;
            width: auto;
        }
        .redbox {
            background-color: rgb(157, 56, 56);
            color: white;
            height: 100px;
            width: auto;
        }
        .amazing {
            background-color: #5b128f;
            color: white;
        }
    </style>
</head>
<body>
    <div>
        <p class="yellowbox">Microsoft is <span class="amazing">amazing!</span>.</p>
    </div>
    <div>
        <p class="redbox">Apple.</p>
    </div>
</body>
```



2.5 Box Model (Padding, Border, Outline, Margin)



If we open our HTML document in the web browser and open the web browser's developer tool, if we navigate to the Elements tab and select a element in the HTML document we should be able to see the Styles and the box model diagram.

Every element is encapsulated in a box model. The box model is a critical concept to understand with CSS and styling elements.

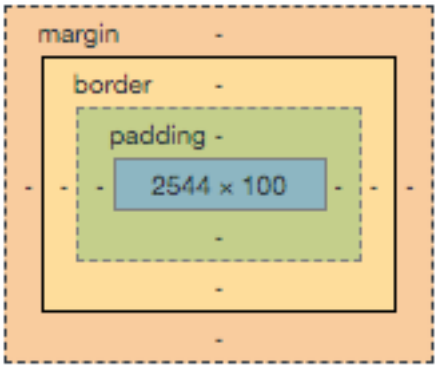
In the above example we have selected the element which wraps the .yellowbox paragraph element.

The blue box in the box model represents the content of the element.

The green box represents the padding which is the distance between the content and the border.

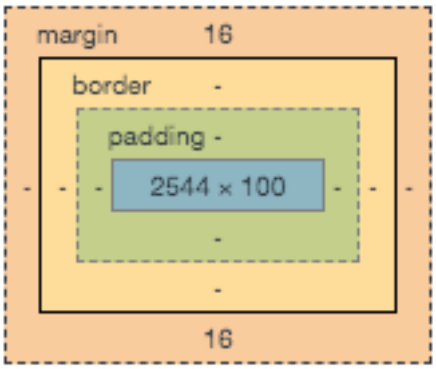
The orange box represents the border.

The peach box represents the margin which separates the HTML element from elements around it.



There is a outline element which is just outside the border but before the margin as highlighted by the thick black border. The outline does not effect the dimensions of the HTML element.

In the above example the padding, border and margin box model for the element has not been set i.e. there are no padding, border or margin for the element.



All HTML elements follow this box model.

If we click on the element inside of the division element, the box model will update to display the calculations of the content, padding, border and margin for the selected paragraph element tag.

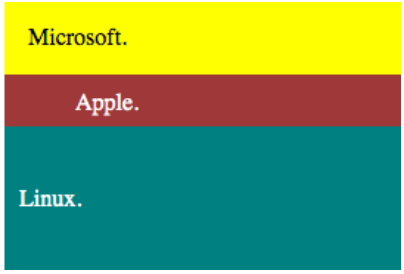
Example on the left is the box model for the paragraph element inside of the division element. The margin for paragraph elements are set by default to 16px.

The padding property is used in the CSS to define a custom padding. There are different ways of defining the padding, for example:

property: value (example)	Description
padding: 16px;	Adds 16px on all sides
padding: 10px 5px 7px 3em;	Adds padding to the top, right, bottom and left side of the content (in that order)
padding: 40px 10px;	Adds padding to the top and bottom and right and left side of the content (in that grouping)

```
.yellowbox {
    background-color: yellow;
    height: auto;
    width: auto;
    padding: 16px;
}
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: auto;
    width: auto;
    padding: 10px 5px 7px 3em;
}
.greenbox {
    background-color: #008080;
    color: white;
    height: auto;
    width: auto;
    padding: 40px 10px;
}
p {
    margin: 0;
}
```

```
<div class="yellowbox">
    <p>Microsoft.</p>
</div>
<div class="redbox">
    <p>Apple.</p>
</div>
<div class="greenbox">
    <p>Linux.</p>
</div>
```



Example of adding padding properties to the elements.

Note: the margin for the elements were set to 0 as the browser default is 16px margin for paragraphs which would add extra spacing to the element.

The border property is used in the CSS to define a custom border. There are different ways of defining the border, for example:

property: value (example)	Description
border: 10px orange ridge;	Defines the border width colour and style (in that order)
border-width: 10px 15px 5px 10px;	Adds border-width value to the top, right, bottom and left side (in that order). Border width is not visible without the border-style property defined
border-width: 10px 20px;	Adds border-width value to the top and bottom and right and right (in that grouping). Border width is not visible without the border-style property defined
border-color: black green orange yellow;	Defines the colour of the border-width from top, right, bottom and left (in that order). Border colour is not visible without the border-width and border-style properties defined
border-style: ridge;	Defines the style of the border e.g. ridge, dashed, dotted, double, groove, etc.
border-radius: 10px	Rounds the edges of the border

```
.yellowbox {
    background-color: yellow;
    height: auto;
    width: auto;
    padding: 16px;
    border: 10px orange ridge;
}
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: auto;
    width: auto;
    padding: 10px 5px 7px 3em;
    border-width: 10px 15px 5px 10px;
    border-color: black green orange yellow;
    border-style: ridge;
}
.greenbox {
    background-color: #008080;
    color: white;
    height: auto;
    width: auto;
    padding: 40px 10px;
    border-width: 10px 20px;
    border-color: paleturquoise;
    border-style: groove;
    border-radius: 10px;
}
p {
    margin: 0;
    background-color: rgba(0, 0, 0, 0.55);
}
```

We can create a circle border element by adding values to the height and width and a unified padding and border-width and a large border-radius value as seen below:


```
.greenbox {
    background-color: #008080;
    color: white;
    height: 100px;
    width: 100px;
    padding: 10px;
    border-width: 10px;
    border-color: paleturquoise;
    border-style: groove;
    border-radius: 80px;
}
```

An outline is a line that is drawn around the HTML element. Unlike borders the outline is not taken into account in the dimensions of the element. Outlines are used to make elements stand out.

property: value (example)	Description
outline-style: solid;	Defines the outline style e.g. dashed, dotted, double, groove, solid, etc.
outline-color: purple;	Defines the colour for the outline
outline-width: medium;	Defines the outline width e.g. medium, thick, thin, etc.

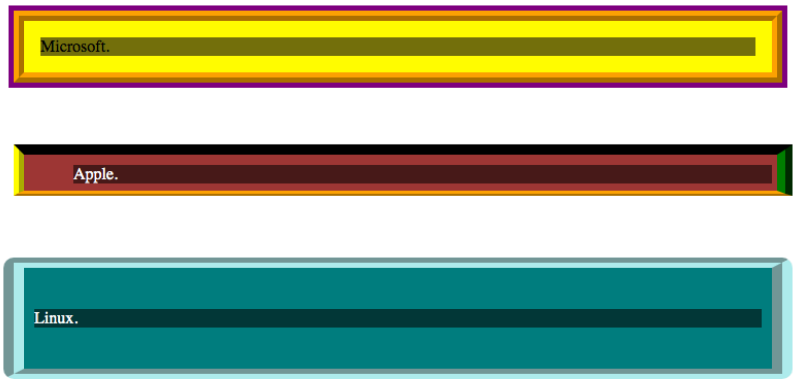
```
.yellowbox {
    background-color: yellow;
    height: auto;
    width: auto;
    padding: 16px;
    border: 10px orange ridge;
    outline-style: solid;
    outline-color: purple;
    outline-width: thick;
}
```

Note: The outline is not taken into account in the dimensions of the element.

Margin allows us to add a gap between itself and any other elements outside the element. To set the margin on an element we would use the margin property. There are different ways to set the margin style:

property: value (example)	Description
margin: 20px;	Adds 20px on all sides
margin: 60px 10px 30px 20px;	Adds margin to the top, right, bottom and left side (in that order)
padding: 30px 10px;	Adds margin to the top and bottom and right and left side (in that grouping)

```
.yellowbox {
    background-color: yellow;
    height: auto;
    width: auto;
    padding: 16px;
    border: 10px orange ridge;
    outline-style: solid;
    outline-color: purple;
    outline-width: thick;
    margin: 20px;
}
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: auto;
    width: auto;
    padding: 10px 5px 7px 3em;
    border-width: 10px 15px 5px 10px;
    border-color: black green orange yellow;
    border-style: ridge;
    margin: 60px 10px 30px 20px;
}
.greenbox {
    background-color: #008080;
    color: white;
    height: auto;
    width: auto;
    padding: 40px 10px;
    border-width: 10px 20px;
    border-color: paleturquoise;
    border-style: groove;
    border-radius: 10px;
    margin: 60px 10px;
}
```



The distance between the first and second element will be the sum of the margin-bottom of the first element and the margin-top of the second element.

2.6 Background

To add a background image using CSS we can use the background-image property. The value is the location of the image which we define within the url( ) property. This is the relative path to the image from the HTML file.

We would want to scale the image so that it fits within the element content area. We can achieve this by using the background-size property. We can set the width and height using percentages.

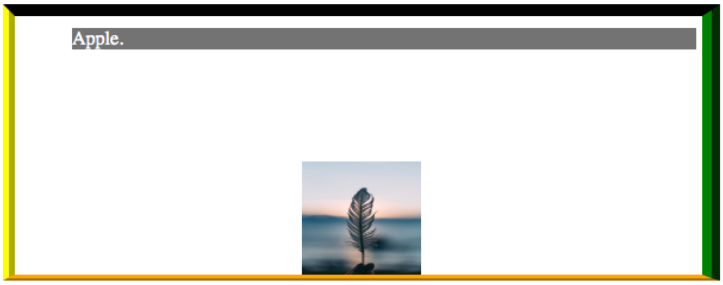
To prevent the picture from repeating itself we can set the background-repeat property to no-repeat. There are other options we can choose for repeat for example repeat-x and repeat-y.

The background-origin property by default is set to border-box i.e. the image would start from the border. We can change this value and there are different options for example content-box which will start the image from the content (in the below the content-box is from the element). All the dimensions are going to be related of the content box-model area for the element.

```
.yellowbox {
    background-image: url(img/image.jpeg);
    background-size: 100% 100%;
    background-repeat: no-repeat;
    background-origin: content-box;
    height: 200px;
    width: auto;
    padding: 16px;
    border: 10px orange ridge;
    outline-style: solid;
    outline-color: purple;
    outline-width: thick;
    margin: 20px;
}
```

We can use the background-position property tells the browser where the image should be placed. There are different options available to choose from e.g. left, right, top, bottom or center of the element. This will centre the element at the position, in the below example the image is centred at the bottom.

```
.redbox {
    background-image: url(img/image.jpeg);
    background-size: 100px 100px;
    background-repeat: no-repeat;
    background-origin: border-box;
    background-position: bottom;
    color: white;
    height: 200px;
    width: auto;
    padding: 10px 5px 7px 3em;
    border-width: 10px 15px 5px 10px;
    border-color: black green orange yellow;
    border-style: ridge;
    margin: 60px 10px 30px 20px;
}
```



2.7 Floating

Float is a property that HTML elements can acquire. This allows elements to float horizontally i.e. to the left or the right (not vertically).

The floating element will float relatively to the containing element for example the division element will float relatively to the body element while the paragraph element will float relatively to the division element.

A element can be floated left, right, centre or none. In the below example, the division element is floated to the left which means floating left relatively to the body element. The element will float as far as possible to the body element. The next element will sit next to the floating element.

```
.yellowbox {
    background-color: yellow;
    height: 200px;
    width: auto;
    float: left;
}
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: 200px;
    width: auto;
}
```


```
.yellowbox {
    background-color: yellow;
    height: 200px;
    width: auto;
    float: left;
}
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: 200px;
    width: auto;
    float: left;
}
.greenbox {
    background-color: #008080;
    color: white;
    height: 200px;
    width: auto;
}
```

Note: The element takes as much space it needs when it floats and the element becomes a inline element. The block element will sit next to the floating element and will take as much space as it needs.

```
.yellowbox {
    background-color: yellow;
    height: 200px;
    width: auto;
    float: left;
}
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: 200px;
    width: auto;
    float: right;
}
.greenbox {
    background-color: #008080;
    color: white;
    height: 200px;
    width: auto;
}
```

Note: interestingly the Linux division element is now sitting between Microsoft and Apple division elements.

The reason for this behaviour is because Microsoft is the first element which is floating to the left. Apple is the next element floating to the right. Linux which comes after Apple, the first available space available is from Microsoft.
Linux which is not a floating element will take as much space it needs which goes from the end of Microsoft all the way to the beginning of Apple which is floated to the right.
To avoid this behaviour we can use the clear property and set its value to both which will prevent any floating elements around the element, resulting in the below example:

```
.yellowbox {
    background-color: yellow;
    height: 200px;
    width: auto;
    float: left;
}
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: 200px;
    width: auto;
    float: right;
}
.greenbox {
    background-color: #008080;
    color: white;
    height: 200px;
    width: auto;
    clear: both;
}
```

2.8 Positioning

Positioning means elements are sitting at a certain location for the HTML element. We can define how these elements are going to behave in the law of the webpage. There are four types of position:

Position Type	Description
static	The default position to the static position i.e. behaving to the natural flow of the webpage
fixed	Element positioned at a fixed position. The position can be defined further using the top and left properties. All other elements will ignore the fixed element in the natural flow of the webpage
relative	Element positioned relative to its original position. The position can be defined further using the top and left properties. Unlike the fixed position, the element reserves its original position in the flow and all other elements around it are still behaving as if the relative positioned element is still in its original position.
absolute	Element positioned relative to the first parent element that is not static. If there are no non-static element then the element will be positioned relative to the <html> element. The position can be defined further using the top and left properties.
top	Defines the top value of a fixed position element i.e. how far away from the top of the web browser page

Position Type	Description
left	Defines the left value of a fixed position element i.e. how far away from the left of the web browser page
z-index	Repositions element positions i.e. which elements appear in-front. The greater the index number will display the image on top of the other elements that have a smaller index.

```
.yellowbox {
  background-color: yellow;
  height: 200px;
  width: auto;
  position: fixed;
  top: 30px;
  left: 30px;
}
.redbox {
  background-color: rgb(157, 56, 56);
  color: white;
  height: 200px;
  width: auto;
  position: relative;
  top: 20px;
  left: 20px;
}
.greenbox {
  background-color: #008080;
  color: white;
  height: 200px;
  width: auto;
  position: absolute;
  top: 40px;
  left: 40px;
}
```


```
.yellowbox {
  background-color: yellow;
  height: 200px;
  width: auto;
  position: fixed;
  top: 30px;
  left: 30px;
  z-index: 1;
}
```

Note: Elements with a positive z-index will appear on top while negative -index will appear below. If an element is not given a z-index it will inherit from its parent. The <html> </html> element's z-index by default is 0.

If a element has position properties while all other elements do not then the positioned element will sit on top of all non-positioned elements. For example, if the Apple element does not have a position property applied whereas both the Microsoft and Linux elements do then the Apple element will be displayed behind the Microsoft and Linux elements.

```
.yellowbox {
  background-color: yellow;
  height: 200px;
  width: auto;
  position: fixed;
  top: 30px;
  left: 30px;
  z-index: 1;
}
.redbox {
  background-color: rgb(157, 56, 56);
  color: white;
  height: 200px;
  width: auto;
}
.greenbox {
  background-color: #008080;
  color: white;
  height: 200px;
  width: auto;
  position: absolute;
  top: 40px;
  left: 40px;
}
```

If elements are all positioned but no z-index property applied to any elements then the last element will always sit on top of the previous element. For Example, the Linux element will sit on-top of Apple element and Apple element on-top of Microsoft element.

```
.yellowbox {
  background-color: yellow;
  height: 200px;
  width: auto;
  position: fixed;
  top: 30px;
  left: 30px;
}
.redbox {
  background-color: rgb(157, 56, 56);
  color: white;
  height: 200px;
  width: auto;
}
.greenbox {
  background-color: #008080;
  color: white;
  height: 200px;
  width: auto;
  position: absolute;
  top: 40px;
  left: 40px;
}
```