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.

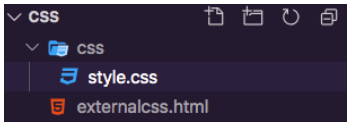
Important Disclaimer: Internal CSS is not recommended as best practice. However, it is important to understand that this is also a possibility for styling a HTML document.

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.

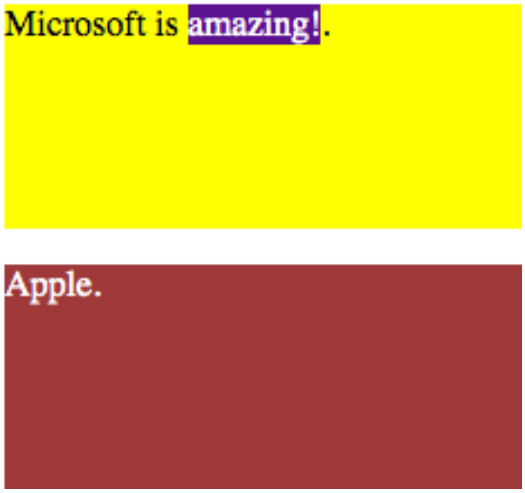
2.4 Div & Span

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.

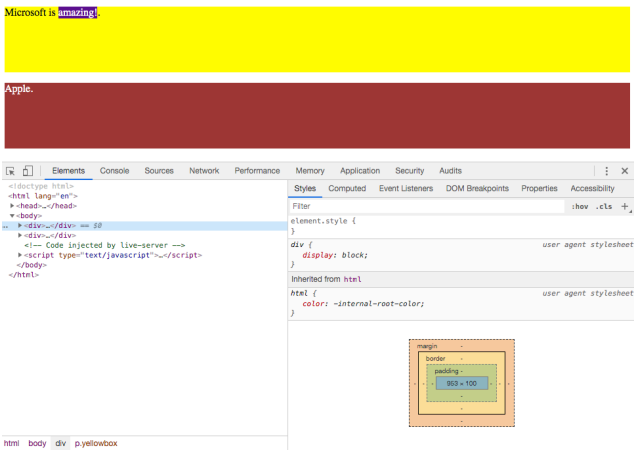
A element tag is an inline element which is used to group inline elements and style them using CSS.

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.

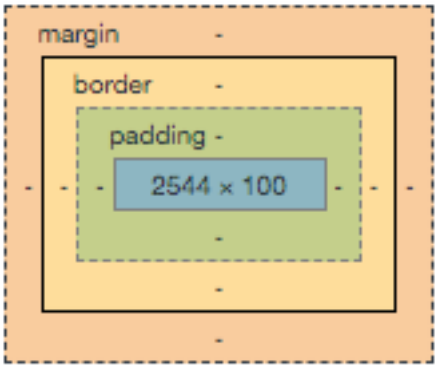
In the above example we have selected the element which wraps the .yellowbox paragraph element.

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.

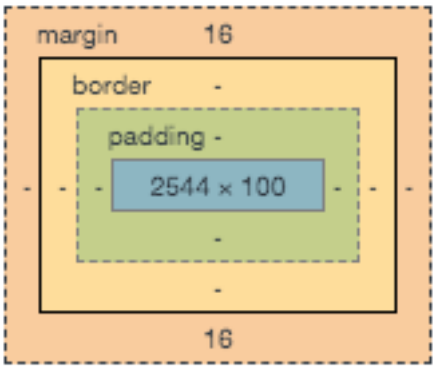
The orange box represents the border.

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.

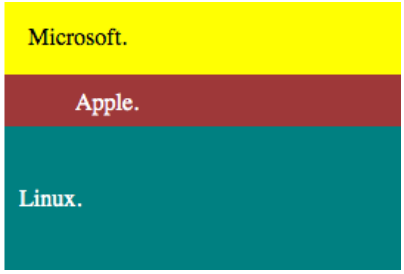
Example on the left is the box model for the paragraph element inside of the division element. The margin for paragraph elements are set by default to 16px.

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.

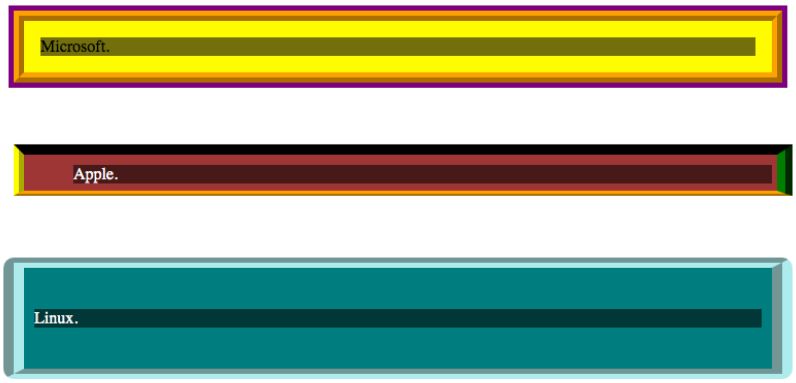
```
.yellowbox {
    background-color: yellow;
    height: auto;
    width: auto;
    padding: 16px;
    border: 10px orange ridge;
    outline-style: solid;
    outline-color: purple;
    outline-width: thick;
}
```

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.

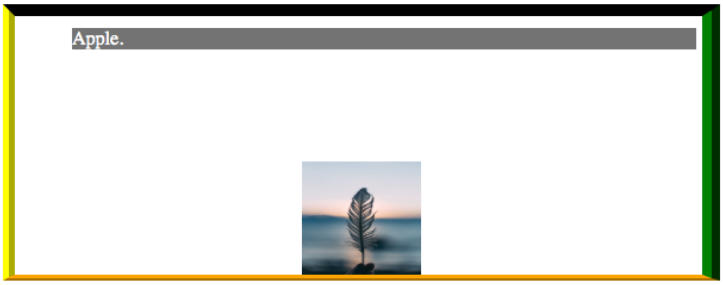
To prevent the picture from repeating itself we can set the background-repeat property to no-repeat. There are other options we can choose for repeat for example repeat-x and repeat-y.

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;
}
```

2.9 Display

The display properties allows you to change the display of a property for example hiding the visibility of an element while preserving the space it was taking up.

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


```
.yellowbox {
  background-color: yellow;
  height: 200px;
  width: auto;
  display: none;
}
```

The display property set to none allows us to hide an element without the element preserving the space it was taking up.

You can also use the display element to change the way inline elements are displayed, for example displaying an inline element as if it was a block element.

```
.amazing {
  background-color: #5b128f;
  color: white;
  display: block;
}
<div class="yellowbox">
  <p>Microsoft is <span class="amazing">amazing!</span>.</p>
</div>
```

Note: The element is displayed as a block element but is continues to be an inline element which means that the element is not allowed to have other block elements inside of it. The same can be done to block elements to display them as inline element.

2.10 Text Decorations (Text Align, Text Font and Text Effects)

There are multiple properties to decorate text using the text-decorations property:

property: value (example)	Description
text-decoration: underline;	Set the text decoration e.g. underline, overline, solid, wavy, etc.
font-weight: 100;	Set the font weight using a numeric value or text e.g. 100 - 900, bold, bolder, lighter, normal, etc.
font-style: italic;	Set the font style e.g. normal, italic, oblique, etc.
font-size: 1.5em;	Set the size of the text
letter-spacing: 10px;	Set the spacing between each letter
text-shadow: 3px 3px 3px green;	Set text shadow using four values: x-shadow, y-shadow, blur radius optional) and colour (optional). By default the colour is black
text-indent: 40px;	Set an indentation for the text (i.e. like a tab/space before the start of a text)

Note: 1em = 16px. Therefore 1.5em is equivalent to 24px (16px + 8px).

The text-align property allows us to align text rendered on the webpage. There are different options/values to choose from. These options are similar to what we would find in Microsoft Word application:

value (example)	Description
left	Align text to the left
right	Align text to the right
center	Align text to the centre
justifu	Justify text
inherit	Inherit text-align property from the parent element

The text-font property allows us to set the font for our text. There are multiple options/ values to choose from:

property: value (example)	Description
font-family: sans-serif;	Select generic font family for the text e.g. cursive, fantasy, monospace, sans-serif, serif, etc.
font-family: Georgia, “Time New Roman”, Times, serif	Set the font-family to a specific font-family. Fonts with spaces in their name require to be wrapped in quotations marks. Multiple font family can be selected as fallbacks including the generic font-family, each separated with a comma

When a element has a very long text, the text will overflow its container i.e. the division element. The overflow of the text can be controlled using text effects properties to display the text inside or outside the division element. There are multiple text effect properties:

property: value (example)	Description
overflow: hidden;	Change the overflow property of text e.g. auto, hidden, scroll, visible, etc.
white-space: no-wrap;	Change the wrap property of text e.g. normal, nowrap, pre, pre-line, etc.
text-overflow: ellipsis;	Change the overflow of text so that it displays … at the end to indicate a overflow of text that has been cut off. There are also other options/values e.g. clip, unset, etc.
word-break: break-all;	Break the word line between letters of long words. Other options available e.g. keep-all, normal, etc.
word-wrap: break-word;	Wrap the word in the container and break the word line between letters of long words. Other options available e.g. normal

Note: we can use the pseudo class to change the styling of a element for example on a hover as seen in the example code to the right. This allows us to change the style of the element i.e. allow the text to overflow and apply no text-wrap when we hover over the element.

2.11 Image Sprites & Image Opacity

Image sprites allows to avoid uploading many pictures at a time. Instead, a master image is used and with the use of CSS positions and dimensions of the image select a chunk from the larger image. This helps avoid multiple HTTP requests and flickering of pictures.

We would first need a <img></img> element tag in the HTML file with an id or class attribute to target in our CSS.

The width and height property will be used to select the dimensions of the sprite image.

The background property is used to select a master sprite image using the url( ) property value. We also pass in the coordinates for the position to start from in the master image. The 0 0 coordinates represents the x and y axis values accordingly. We would use negative numbers to move the start position.

To calculate the start position we would need know the dimension of the image for example if you have a 2248 x 1500 size image. Therefore, a coordinate of (0, 0) is the same as saying (0+2248, 0+1500) i.e. 2248px, 1500px will also work.
If we choose a coordinate of (-310, -304) this is the same as (-310+2248), (-304+1500).

The width and height property will display the image at those dimensions from the coordinate starting position. This allows us to select an image from a master sprite image.

```
    <style>
        #image1 {
            width: 100px;
            height: 100px;
            background: url(img/image.jpeg) -500px -1000px;
        }
    </style>
</head>
<body>
    <img id="image1">
</body>
```

Image sprites are a powerful tool and is something to keep in mind when doing web design and maintaining web design.

The opacity property allow us to define the transparency of an image. We can use any value between 0 and 1 to define the opacity level.

```
        #image1 {
            width: 100px;
            height: 100px;
            background: url(img/image.jpeg) -500px -1000px;
            opacity: 0.4;
        }
```

2.12 Styling Lists and Links

The list-style property allows you to style lists elements and there are various options/ values to choose from for example: armenian, circle, decimal, decimal-leading-zero, disc, georgian, square, etc.

The list-style-image property allows you to style the list bullets using an image which is specified using the url( ) property value.

We can style and control where the picture appears and position the picture the way we want.

First we need to style the unordered list element and set the list-style to none to remove the default list items markers and set the margin and padding to 0 for cross-browser compatibility.
Second we would want to target the <li> items in the list and add a background-image property and select an image using the url( ) property value. This causes the image repeating itself. Therefore, using the background-repeat property and setting this to no-repeat will correct the issue.
Finally, we would set some padding to position the text so that it does not overlap the image. The background-position property allows us to set the position of the image on both the x (horizontal) and y (vertical) axis.

The default browser style for links is blue underlined text for non-visited links. When you visit a link this turns to a purple underlined text. To change the default style for all links on a webpage is to target the a element tag. We can then target the various pseudo links type for example:

tag: pseudo (example)	Description
a:link { }	Target non visited links
a:hover { }	Target hovered links style
a:active { }	Target activated links style i.e. when link is clicked
a:visited { }	Target visited links style. There are some limitations to a visited link style such as the font-size depending on the browser

Note: We can target specific elements that contains a link element by chaining the style target for example div.yellowbox a:link { …; }.

2.12 Gradients

Gradients allow you to create a smooth transition between different colours. There are different types of gradients i.e. linear gradient and radial gradient.

Disclaimer: Gradients do not work the same in all browsers and therefore may require a special syntax appended to the property value to make them compatible. You would need to duplicate the same property to cover all browsers for cross compatibility:

-webkit- = Safari Web Browser
-moz- = Mozilla Firefox Web Browser
-o- = Opera Web Browser

property: value (example)	Description

background: linear-gradient(blue, white);
 	
background: linear-gradient(to left, blue, white);
 	
background: -webkit-linear-gradient(blue, white);
	A linear gradient between two or more colours. By default the gradient goes from top to bottom
	
	The default direction can be changed by entering the direction as the first argument followed by the colours. There are multiple directions e.g. to left, to bottom left, to right, etc.
	The to left direction will gradient the colours from the right to left i.e. blue on the right side transitioning to the white on the left side
	
	The original direction can be rotated anti-clockwise using degree e.g. 90deg
	
	Appending the special syntax makes the linear-gradient style property compatible with a certain browser that do not support the original syntax

background: radial-gradient(blue, white) background: radial-gradient(circle, blue, white) background: radial-gradient(blue 5%, white 10%) background: -moz-radial-gradient(blue, white)	A radial gradient between two or more colours. By default the gradient goes from the centre out The first argument can specify the radial shape to be a circle before applying the colour arguments Adding a percentage after the colour will customise the space each colour will take using a percentage. The total percentage must add unto 100% Appending the special syntax makes the radial-gradient style property compatible with a certain browser that do not support the original syntax

2.13 2D and 3D Transforms

2D and 3D transforms allow us to change the shape of some elements as well as their size and positioning.

For 2D transformation the syntax starts with transform: followed by choosing the property to change/transform for that element. There are many 2D transforms we can choose from.

Disclaimer: Transforms do not work the same in all browsers and therefore may require a special syntax appended to the property value to make them compatible. You would need to duplicate the same property to cover all browsers for cross compatibility:

-webkit- = Safari Web Browser
-moz- = Mozilla Firefox Web Browser
-o- = Opera Web Browser

property: value (example)	Description
transform: rotate(30deg); -webkit-transform: rotate(30deg);	Rotate the element by a degree. Positive number for clockwise and negative number for anti-clockwise Appending the special syntax makes the transform property compatible with a certain browser that do not support the original syntax
tranform: translate(10px, -30px) -moz-transform: rotate(30deg);	Translates (moves) the element on the x and y axis (in that order). Positive number to move right/down and negative number to move left/up Appending the special syntax makes the transform property compatible with a certain browser that do not support the original syntax
transform: scale(1, 0.5);	Increase/decrease the size of the element. One value will increase/decrease both the width and height of the element by the same value else the first value is for the width and second value for the height. Value of 1 is the original size of the element while a smaller/larger number will decrease/increase the scale

transform: skewX(10deg);
transform: skewY(10deg);
 	
transform: skew(10deg, 10deg);
	Skewing the object by a degree either on the X or Y axis depending on the property value selection
	
	The skew property can be used for skewing the element on both the x and y axis at the same time (in that order)

3D transforms allows us to transform the element in three dimensions i.e. the x, y and z axis. This allows us to create 3D transform and add depths to the elements. The syntax is similar to the 2D transform i.e. starts with transform: followed by the 3D transform property values:

property: value (example)	Description
transform: rotateX(30deg); -webkit-transform: rotateX(30deg);	Rotate the element by a degree on the X axis. Positive number for clockwise and negative number for anti-clockwise Appending the special syntax makes the transform property compatible with a certain browser that do not support the original syntax
transform: rotateY(30deg); -moz-transform: rotateY(30deg);	Rotate the element by a degree on the Y axis. Positive number for clockwise and negative number for anti-clockwise Appending the special syntax makes the transform property compatible with a certain browser that do not support the original syntax
transform: rotateZ(30deg); -o-transform: rotateZ(30deg);	Rotate the element by a degree on the Z axis. Positive number for clockwise and negative number for anti-clockwise Appending the special syntax makes the transform property compatible with a certain browser that do not support the original syntax

2.14 Transitions

Transitions is used to create effects for HTML elements when they move from one style to another. The transition property is used to define what property would change and the duration for the transition.

For the transition to take effect we would need to use the pseudo element such as :hover to change the property of the element. This would then trigger the transition whenever we hover over an element. It is important to change the property specified in the transition property.

```
.yellowbox {
    background-color: yellow;
    height: 200px;
    width: 200px;
    transition: width 1s;
}
.yellowbox:hover {
    width: 400px;
}
```


We can create more complex transitions with multiple properties to transition from the original state to the changed state as seen in the example on the right.

```
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: 200px;
    width: 200px;
    transition: width 1s, height 1s, transform 1s;
}
.redbox:hover {
    width: 400px;
    height: 400px;
    transform: rotateX(180deg);
}
```

2.15 Animation

The animation property is used to create animations and is added to the element which will use the animation. This property takes in several property values: animation name, duration, time-function, delay, iteration-count, direction and fill mode. The first two arguments are mandatory values.

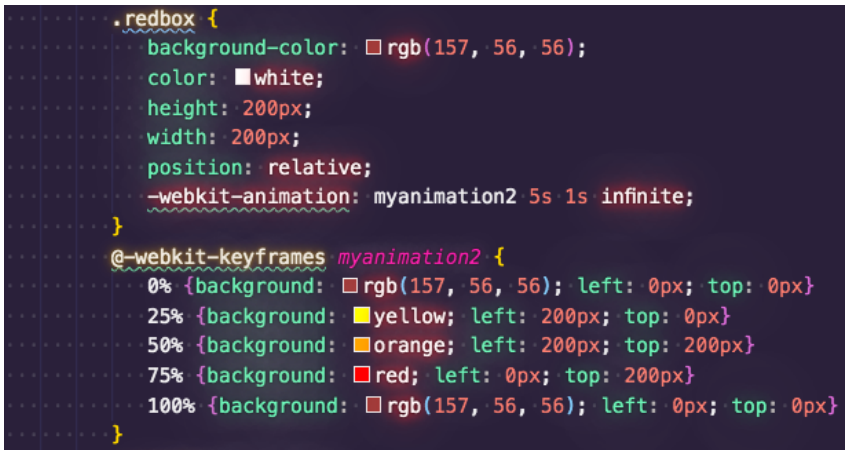
The animation would then need to be defined outside of the element using @keyframes followed by the name of the animation. The keyframe defines the original from style to the new to style for the animation.

```
.yellowbox {
    background-color: yellow;
    height: 200px;
    width: 200px;
    animation: myanimation 5s;
    -webkit-animation: myanimation 5s;
    -moz-animation: myanimation 5s;
    -o-animation: myanimation 5s;
}
@keyframes myanimation {
    from {
        background: yellow;
    }
    to {
        background: orange;
    }
}
```

Disclaimer: Animations syntax does not work the same in all browsers and therefore may require a special syntax appended to the property value to make them compatible. You would need to duplicate the same property to cover all browsers for cross compatibility. This appended syntax needs to be appended to both the animation and keyframe properties:

-webkit- = Safari Web Browser
-moz- = Mozilla Firefox Web Browser
-o- = Opera Web Browser

The alternative syntax is to define the animation CSS style at different keyframes indicated by the percentage of time. In the below example the animation has a 1 second delay and a infinite loop provided as optional arguments to the animation property.



```
.redbox {
    background-color: rgb(157, 56, 56);
    color: white;
    height: 200px;
    width: 200px;
    position: relative;
    -webkit-animation: myanimation2 5s 1s infinite;
}
@-webkit-keyframes myanimation2 {
    0% {background: rgb(157, 56, 56); left: 0px; top: 0px}
    25% {background: yellow; left: 200px; top: 0px}
    50% {background: orange; left: 200px; top: 200px}
    75% {background: red; left: 0px; top: 200px}
    100% {background: rgb(157, 56, 56); left: 0px; top: 0px}
}
```

END