Advanced CSS

Section 2: Advanced CSS

2.1 Backgrounds

In CSS there are several background properties which are:

Property	Description
background-color	Change the colour of the background
background-image	Set an image as a background using the url( ) pointing to the image path
background-size	Change the size of the image where the first value is for the width and second values for the height of the image. The image can lose its quality when changing the size of the image using both values and so recommended to use one value and the other as auto for its value
background-repeat	The value of no-repeat; will stop the default background-image behaviour of repeating the image horizontally and vertically. The value of repeat-x; will repeat the image horizontally while the value repeat-y; will repeat the image vertically
background-position	Control the position of the background-image and control what part of the image is shown. Values that can be used are px, %, em, rem, top, right, bottom, centre, etc. This can take two values where the first value is the position from the left and the second value is the position from the top. Negative values can be used to change in the opposite directions
background-attachment	This can be used to create a parallax effect. The fixed; value will change the position of the image according to the viewport. The background image will remain fixed in its position when scrolling down the page (this is known as the parallax effect)
background-origin	Specifies where to start displaying the background image and by default is set to padding-box; i.e. the image starts displaying from the top left corner of the element's padding edge. The content-box; value the image starts displaying after the padding at the top left corner of the elements content edge. The border-box; value the image starts displaying at the top left corner of the element's border
background-clip	The default value is border-box; and the image will display from the top left corner of the outer edges of the element's border. The padding-box; value displays the image from the top left corner of the outer edges of the element's padding. The content-box; value displays the image from the top left corner of the outer edges of the element's content.

Property	Description
background	Allows to set multiple different background properties mentioned above for example background: green url(img/img.png) no-repeat center fixed /cover; When defining the background-size property in the short form this requires the background-position to be defined and must come after it using a forward slash (/) as a requirement to work. Multiple image can be styled using this one background declaration each separated with a comma (,)

Important: If the name of the image or the path to that image contains a space then we must wrap the file path of that image in double quotation marks (“ ”). We can use double quotes in any case regardless whether there is a space or no space in the image name or file path as this will continue to work.

The default behaviour for background-image is to repeat the image horizontally and/or vertically to fit the image inside the box size of the element.
The image is placed from the top left hand corner and exceeds the size of the box then by default the image will be clipped on the right and bottom sides.
If we use the declaration of background-position: right bottom; then the image will start from the bottom right corner and will be clipped on the left and top sides. This declaration can take up to 4 values if using the text value along with measurements for example background-position: right 50px bottom 50px which will move the starting position from the right and bottom corner by 50px on both sides.

We can control the clipped sides by using percentage values for the background-position property. If we imagine the clipped sides are 100% at the right and 100% at the bottom. If we want to clip the image equally on all four sides we would need to use 50% for the first value and 50% for the second value to the background-position property. This will result in equal clipping on the top and bottom sides and left and right sides. This will centre the image but we can also use the center value to centre the image without percentage values.
In the example of background-image: 90% 40%; the image would be clipped by 90% on the left and 10% on the right while its is clipped 40% from the top and 60% at the bottom. This is how using percentage values differentiates from the other measurement values when using it with the background-image property.

The background-size allows us to change the size of the image using two values but this will also have an effect on the image quality. There are text values we can use such as cover; which will cover the entire element without losing the image quality but the image will be clipped. The contain; value which acts like the value of 100% where the image will take the full width of the element and the height is set automatically. This also does not lose the value of the image. Out of the two property values cover is more typically used.

Important: Images do not take into account the width of the parent element until we define the width manually for the image by selecting the image element.

2.2 Gradients

Gradients allow us to display multi-transitions between two or more colours as well as define directions for those transitions.

```
background: linear-gradient(rgba(0, 255, 0, .1), rgba(0, 255, 0, .2)), url("img/image 1.jpeg") no-repeat center /cover;
```

The linear-gradient property is used as a value within the background property. The first argument to linear-gradient is the first colour and the second argument is the second colour. This will create a smooth transition from top to bottom between the two colours which is the default behaviour (i.e. to bottom).

Before the colour arguments for the first argument we can select the direction of the transition by using to followed by the direction e.g. to right. If this argument is omitted then the default to bottom direction is used. To make the direction go horizontally you must define the horizontal and vertical directions for example “to right bottom” will start the gradient from the top left hand corner transitioning to the bottom right corner.

```
background: linear-gradient(to right, rgba(0, 255, 0, .1), rgba(242, 255, 0, 0.6)), url("img/image 1.jpeg") no-repeat center /cover;
```

We can mix more than two colours by specifying as many colours as additional arguments.

Images can also be added to the background shorthand declaration and this will apply the gradient colours on top of the background image. We can omit the background-image and this will display the gradient colours only.

Disclaimer: There is also a radial-gradient property but is not typically used when designing websites.

2.3 Shadows

Shadows can be added to both text and box elements.

```
text-shadow: 3px 3px 10px #888;
```

The text-shadow property is used to set the shadow of text elements. This property takes in a number of values. The first indicates the position of the shadow in a horizontal line, the second indicates the position of the shadow vertically. These are the two mandatory values required. The third value defines how blurry the shadow should be. The last value defines the colour of the shadow (the default colour of shadow is black).

```
box-shadow: 10px 10px 20px #aaa;
```

The box-shadow property is used to set the shadow of box elements and this is the main difference between text-shadow and box-shadow. The box-shadow takes in the same values as text-shadow.

2.4 Transitions

CSS transitions property allows you to change any of the CSS properties for an element from one value to another smoothly over a specified duration. There are four transitions properties:

Property	Description
transition-property	Defines the property to add a transition effect to
transition-duration	Defines how long the transition effect should to complete
transition-delay	Defines the duration of the delay before the transition effect starts
transition-timing-function	Defines the different speed of the effects when the transitions happens
transition	The shorthand property syntax for the four properties above

Important: The transition-property and transition-duration declarations are mandatory for transitions to work.

A pseudo element is also required to transition to.

By default the transition-timing-function is set to ease but can be changed to other timing-functions such as ease-in, ease-out, ease-in-out, linear, etc.
The cubic-bezier function value allows you to define your own speed of transition effects manually. The cubic-bezier function takes in 4 arguments. The following website is a useful resource for creating and viewing timing-functions (https://cubic-bezier.com/).

To transition multiple properties with different duration we must specify the four values for each property which are separated with a comma (,) as seen in the example on the right.
To transition all properties with the same duration, delay and timing-function we can use the 'all' value for the transition property value as seen in the above example.

2.5 Transform

Transformation is an effect which allows to change an element's position, size, rotate and shape either as a 2D or 3D transformations. Below are the various transform properties available to use:

Property: Value; (example)	Description

transform: translateX(100px);
transform: translateY(200px);
transform: translate(100px 200px);
	Position the element on the horizontal X axis
	Position the element on the vertical Y axis
	Shorthand transform syntax property on both X axis, Y axis

transform: rotate(90deg); transform: rotate(-90deg);	Rotate the element by a certain degree value clockwise A negative number rotates the element counter-clockwise
transform: scaleX(2); transform: scaleY(.5);	Increase/decrease the size of an element's width Increase/decrease the size of an element's height

transform: scale(1, 1);
transform: scale(2, 2);
transform: scale(.5, .5);
	Shorthand scale syntax to increase/decrease the size of an element width (X) and height(Y).
	By default the width and height of an element is (1, 1)

transform: skewX(30deg);
transform: skewY(30deg);
transform: skew(30deg, 30deg);
	Skew the element on the horizontal X axis by a degree
	Skew the element on the vertical Y axis by a degree
	Shorthand skew syntax property on both X axis, Y axis

transform: rotateX(180deg);
transform: rotateY(180deg);
transform: rotateZ(180deg);
	3D transformation of rotating an element on the X, Y or Z axis by a certain degreee

transform: translate(200px, 200px) scale(1.5, 1.5);	Shorthand syntax to combine multiple transformation properties

We can use the transition property along with the transform property to create an animation/transition between the original and pseudo element using purely CSS.

Disclaimer: The transform property can only be applied once on an element and therefore we cannot declare multiple transforms on a single element. To use multiple transform properties on a single element the shorthand transform property syntax must be used and each transform value must be separated with a space.

2.6 Animation

To create animations in CSS we need to use @keyframes. In the keyframes we define the initial style of an element and the new styles the element will get during the animation. When creating a animation we use the @keyframes followed by the animation name.

To define the starting style of the element we would use the from keyword and define the style in the curly brackets. To define the end style of an element we would use the to keyword and define the end style of the element in the curly brackets.

Alternatively, the more efficient way to create animation is to define the styles at various frame. To indicate the frame and its style we define the percentage from 0% to 100% and then define the style within the curly brackets for each style frame. This provides much more greater control over the animation compared to the from and to keywords which only provide the animation style from the start (0%) and the end (100%) only.

Once the @keyframe has been defined the animation can be connected to the element by using several different animation properties on the target element itself. Below are the various animation properties available:

Property: Value; (example)	Description
animation-name: anim;	Select by name of the @keyframe animation style to apply to the element (this should always match)
animation-duration: 4s;	Set the duration of the animation (by default this is set to 0)
animation-delay: 2s;	Defines the start delay of the animation
animation-iteration-count: 1; animation-iteration-count: infinite;	Control how many times the animation should loop. By default this is 1 but can be changed to any number or infinite
animation-direction: normal; animation-direction: reverse; animation-direction: alternate; animation-direction: alternate-reverse;	Allows to run the animation in different directions. There are a total of 4 options available: normal, reverse, alternate, and alternate-reverse. By default the direction is set to normal. For alternate or reverse-alternate directions to work the iteration-count must be 2 or greater
animation-timing-function: linear;	This property is exactly the same as how it works for transition-timing-function property and takes the same values e.g. ease, ease-in, linear, cubic-bezier( ), etc
animation-fill-mode: none; animation-fill-mode: forwards; animation-fill-mode: backwards; animation-fill-mode: both;	By default this is set to none. The forward value provides a way to stop the element going back (or jump) to its initial position when the the animation ends. The backward value allows the element to start at the start animation position rather than the initial position. The both value allows the use of both forward and backward at the same time.
animation: anim 4s ease-in 2s infinite alternate both	Shorthand syntax to use the different animation properties at the same time. Values should be in the order as seen in the example i.e. name, duration, timing-function, delay, iteration-count, direction and fill-mode

CSS is not a programming language but it allows us to create some nice and pleasing effects using animations.

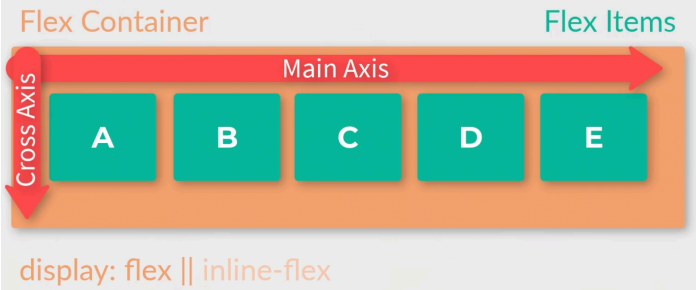
2.7 What is Flexbox?

The most important aspect of CSS is knowing how to align elements on the page in an easy and flexible way. Floats is one way of aligning elements but there are now less popular, inconvenient and requires a lot of effort. CSS now allows us to use CSS Flexbox and CSS Grid modules which are extremely popular, easy & convenient and flexible.

Flexbox is a new and modern technique which improves the way of aligning elements and allows us to avoid floats. Flexbox is fully supported in all modern web browsers.

Flexbox is used to manage alignment of items, directions and order in the container. There exists several properties separately for both the container and its child items which we need to be familiar with.

Relationship between container and its items:



The containers requires us to set the display to either flex or inline-flex.

Note: The inline-flex is not really used in the real world.

The container will then have influence on its child items.

The direction of how elements are laid out in Flexbox are along the main axis which by default goes from left to right and the cross axis which by default goes from top to bottom. Using different properties and values the direction of the main axis and cross axis can be changed.

Below are the various Flexbox properties for both the container and items:

Container Properties: Value (Example)	Description
display: flex display: inline-flex	Required to set the container display property in order to use the other Flexbox properties on both the container and its child element
flex-direction: row; flex-direction: row-reverse; flex-direction: column; flex-direction: column-reverse;	Defines the direction that the container wants to set for its flex-items. As default it is laid out as row. Flex column changes the direction of the main axis to top to bottom and cross axis left to right i.e. opposite to the default row direction
flex-wrap: nowrap; flex-wrap: wrap;	Manage the container to lay out the flex-items in either a single or multiple lines. The nowrap is the default value
flex-flow: row wrap;	Shorthand syntax for combining the direction and wrap properties
justify-content: flex-start; justify-content: flex-end; justify-content: center; justify-content: space-between; justify-content: space around; justify-content: space-evenly;	Align items along the main axis of the container i.e. align items at the beginning, centre or end of the line. Items can also be aligned with spaces between, around or spaces equally
align-items: stretch; align-items: flex-start; align-items: flex-end; align-items: center; align-items: baseline;	Similar to justify-content but allows to align the container's child items on the cross axis direction. By default this is set to stretch

Container Properties: Value (Example)	Description
align-content: stretch; align-content: flex-start; align-content: flex-end; align-content: center; align-content: space-between; align-content: space-around;	Similar to justify-content but is used when there are extra spaces on the cross axis (i.e. wrapped contents). By default this is set to stretch

Item Properties: Value (Example)	Description
order: 0;	Order the numbering of items from 0 to # which allows to reposition elements in the flex-container by their order row/ column number. Negative numbers can also be used. Items with the lowest order number sit in-front of higher order numbers. The ordering of flex-items depend on the flex-direction
align-self: auto; align-self: stretch; align-self: flex-start; align-self: flex-end; align-self: center; align-self: baseline;	Override each individual flex items the default alignment. The property is the same as the flex-container's align-item property but it allows to align the individual flex-item
flex-grow: 0;	Change the size of the flex-item when the size of the viewport increases. By default this is set to 0 which will keep the same size of the item regardless the browser viewport increase or decrease. The number value defines the speed of the growth for the item to take as much space that is available. The item will grow proportionate to the number assigned if multiple items are given the flex-grow declarations with a value greater than 0
flex-shrink: 1;	Manage shrinking items as possible i.e. the opposite to flex-grow. The default is set to 0 i.e. no shrink. The number defines how fast the item should shrink in comparison to other items. The container-property flex-wrap must be set to no wrap because the wrap value does not allow items to shrink
flex-basis: auto;	Defines the length of the flex-item content i.e. the width or height depending on the flex-direction of the container. By default this is set to default but can be overridden with a different value such as 100px. The flex-basis value will override the width value of an item if both properties are declared on that item unless the flex-direction is set to column in which case it will override the height property
flex: 0 1 auto;	Shorthand syntax for the grow, shrink and basis properties

2.8 CSS Media Queries

In general when building websites or web-apps you should keep in mind that it should always be responsive on the various screen devices i.e. phone, tablets (portrait/ landscape) and desktop.

There are two ways to build websites/web apps i.e. mobile first and desktop first approaches. The first builds for the mobile in mind and then scales up/responds to larger device screens while the latter takes the opposite approach of building for the desktop and then scaling down/responds to smaller device screens.

In the browser developer tools within the Elements tab we are able to click on the device icon to change the resolution of the screen by either selecting preset device resolutions or custom width and height resolutions. This allows you to test your webpage/web-app across different screen sizes.

CSS media queries allows us to create CSS rules which should be applied on specific screen sizes. To use the CSS media query this is achieved using the @media( ) keyword. We can use the max-width to decide when the styles in the curly brackets should apply.

In this example, while the width of the pages is greater than a 1000px then the <h1> element will maintain its normal style. However, as soon as the page becomes less than or equal to 1000px then the new style will apply i.e. text colour is colour red.

Important Note: Using min-width does the opposite of max-width.

Disclaimer: The proper break-points to choose will depend on the type of webpage/web app that is being built and how the elements are aligned and it is not a good practice to base this on popular device screen sizes. The developer tool in the browser is a great tool to help decide when to add a breakpoint where the content breaks the aesthetics of the webpage/web app that is being built.

END