General Intro

* css code can be placed inline with html code, in the html file but in a separate section of the same file or in an external file.
* when inline it goes within the relevant tag, when in file it goes in a **<style>** tag in the header section, when external it goes in a .css file
* for external css files you have to place a link to the file in the html: <**link** href="pathToCssFile" type="text/css" rel="stylesheet">
* rel — this attribute describes the relationship between the HTML file and the CSS file. Because you are linking to a style sheet, the value should be set to style sheet.
* to select an HTML element and style it in CSS use the what is inside the HTML tag followed by curly brackets e.g. **p {property: value;}**
* to select a class and style it in CSS use dot then the class name followed by curly brackets e.g. **.className {property: value;}**
* to select a id and style it in css use hash then the id name followed by curly brackets e.g. **#idName {property: value;}**
* IDs are meant to style only one unique element they should be used sparingly and only on elements that need to always appear the same.
* **Specificity** is the order by which the browser decides which CSS styles will be displayed. A best practice in CSS is to style elements while using the lowest degree of specificity, so that if an element needs a new style, it is easy to override.
* IDs are the most specific selector in CSS, followed by classes, and finally, tags. If IDs are used in conjunction with classes and tags the ID css will always override the others.
* you can combine multiple selectors in one css style to be more specific by using **chaining** : **h1.special { property: value;}**
* to select nested html elements first select the mother class then the nested item: **.className h1 {property: value;}**
* you can override *any* style no matter how specific it is if you use the important command. This should only be used in emergency as it will override any other css rule: **p {color: blue !important;}**
* if you have multiple selectors applying the same css styles you can combine them into one comma separated statement: **h1, .menu { font-family: Georgia; }**

CSS Properties

* font-family: indicates the font type. Default value: times new roman. Client needs to have the font installed for it to render. Limit the font types to 2-3 per page for best performance. let the font name be in quotes to cover spaces in the name.
* font-size: indicates the size of the font. It can be set in pixels or em units
* font-weight: indicates how bold the text will appear. It can be bold or normal
* text-align: positions the text with respect to its parent. It can be left, right or centre
* color: this sets the foreground color of an element
* background color: this sets the background color of an element
* text-transform: change text to upper or lowercase
* opacity is the measure of how transparent an element is. It's measured from 0 to 1
* background-image property will set the element's background to display an image. **background-image: url ("images/mountains.jpg");**
* The **box model** comprises the set of properties, which define parts of an element that take up space on a web page.
* width and height — specifies the width and height of the content area.
* padding — specifies the amount of space between the content area and the border.
* border — specifies the thickness and style of the border surrounding the content area and padding. **border: 1px dotted red**
* margin — specifies the amount of space between the border and the outside edge of the element.
* Pixels allow you to set the exact size of an element's box (width and height). When the width and height of an element are set in pixels, it will be the same size on all devices
* a border doesn't have to be square. You can modify the corners of an element's border box with the border-radius property.
* padding property lets you specify exactly how much padding there should be on each side of the content in a single declaration. **padding: 6px 11px 4px 9px;** clockwise rotation from top. This is the same for margins as well.
* To center content use: **width: 400px; margin: 0 auto;** you cannot center without a width specified. the top/bottom margins will be zero and the side margins will be automatically set to center
* Horizontal margins add, so the total space between the borders of adjacent elements is equal to the sum of the right margin of one element and the left margin of the adjacent element.
* Vertical margins collapse, so the space between vertically adjacent elements is equal to the larger margin.
* min-width — this property ensures a minimum width of an element's box.
* max-width — this property ensures a maximum width of an element's box. Similar exist for height as well. All these help in controlling the experience for clients on different devices.
* The overflow property controls what happens to content that spills, or overflows, outside its box. It can be set to one of the following values (i.e. set to the parent of the desired element:
* hidden - when set to this value, any content that overflows will be hidden from view.
* scroll - when set to this value, a scrollbar will be added to the element's box so that the rest of the content can be viewed by scrolling.
* visible - when set to this value, the overflow content will be displayed outside of the containing element. Note, this is the default value.
* The first rule in an external css file should be: **\* {margin: 0; padding: 0; }** it resets the default margin and padding values of all HTML elements to ensure you are working with a clean slate.
* An element with **display: none**will be completely removed from the web page. An element with **visibility: hidden**, however, will not be visible on the web page, but the space reserved for it will.
* The default box sizing model **content-box** comes with sizing issues so it is better to reset the box sizing model in the beginning to border-box which auto sizes your content for you so that dimensions remain accurate. Use: **\* {box-sizing: border-box;}**