**CSS Terminology & Definitions (Answers)**

* 1. **Internal or embedded CSS** requires you to add *<style> tag in the <head>* section of your HTML document. This CSS style is an effective method of styling a single page. However, using this style for multiple pages is time-consuming as you need to put CSS rules on every page of your website. In addition, adding the code to the HTML document can increase the page’s size and loading time.
  2. With **external CSS**, webpages are linked to an external ***.css*** file, which can be created by any text editor. This CSS type is a more efficient method, especially for styling a large website. By editing one .css file, an entire site can be changed at once. Since the CSS code is in a separate document, HTML files will have a cleaner structure and are smaller in size. Furthermore, the same .css file can be used for multiple pages. However, until the external CSS is loaded, pages may not be rendered correctly. Finally, by uploading or linking to multiple CSS files, the site’s download time can be increased.
  3. **Inline CSS** is used to style a specific HTML element. For this CSS style, only the addition of the style attribute to each HTML tag is needed, without the use of selectors. This CSS type is not really recommended, as each HTML tag needs to be styled individually. Website management may become too hard if only inline CSS is used. However, inline CSS in HTML can be useful in some situations. For example, in cases where there is no access to CSS file or single element styles need to be applied. Advantages of Inline CSS include the easy and quick insertion of CSS rules to an HTML page as well as the fact that creating and uploading a separate document as in the external style is not necessary.

1. 1. Class selector syntax: *.className { CSS rules }*
   2. Id selector syntax: *#idName { CSS rules }*
2. By declaring the two different selectors separated by a comma before opening the brackets to write the single rule.
3. .primary, #title { single rule }
4. The **descendant combinatory**– typically represented by a single space (“ “) character – combines two selectors such that elements matched by the second selector are selected if they have an ancestor (parent, parent’s parent, parent’s parent’s parent, etc.) element matching the first selector. Selectors that utilize a descendant combinatory are called descendant selectors.
5. The rule that uses one class selector because a class selector will beat any number of type selectors regarding specificity.
6. Content -> Padding -> Border -> Margin
7. The **box-sizing**CSS property sets how the total width and height of an element is calculated.
8. 1. In the **standard box model**, if you give a box an inline-seize and a block-size (or width and a height) attributes, this defines the inline-size and block-size (width and height in horizontal languages) of the content box. Any padding and border is then added to those dimensions to get the total size taken up by the box.
   2. In the alternative box model. Any width is the width of the visible box on the page. The content area width is that width minus the width for the padding and border. No need to add up the border and padding to get the real size of the box.
9. Margin
10. Padding
11. Margin
12. Block elements begin from a new line by default and cover space to its left and right as far as it can go. The height that it covers is equal to the content height. Also, it covers the whole horizontal space of its parent element.  
      
    Inline elements never start from a new line and only cover the width according to the size of bounded tags in the HTML element.
13. Compared to *display: inline*, the major difference is that *display: inline-block* allows to set a width and height on the element. Also, with *display: inline-block*, the top and bottom margins/paddings are respected, but with *display: inline* they are not.
14. Block
15. Inline
16. Block
17. Inline
18. A **flex container** is any element that has the property **display: flex;** on it, while a **flex item** is any element located inside the **flex container**.
19. By locating an element inside a flex container (an element with the **display: flex;** property on it).
20. The 3 properties are: **flex-grow**, **flex-shrink** and **flex-basis.**
21. By setting the **flex-direction** property to **column.**
22. **justify-content** aligns items across the main axis.  
     **align-items** changes the placement of items along the cross-axis.
23. By setting both **justify-content** and **align-items** to center.
24. **space-between**: items are evenly distributed in the line; first item is on the start line, last item on the end line.  
     **space-around**: items are evenly distributed in the line with equal space around them.