|  |  |
| --- | --- |
|  | Information Technology Department - State Polytechnic of Malang  **Jobsheet-2: CSS**  **Course: Web Design and Programming**  Web Design and Programming Teaching Team  September 2024 |

**Student Identity**

**Name : Lovie Jechonia Tonimba**

**NIM : 244107060101**

**Class : 2G**

**Major : D-IV Business Information System**

**Topic**

* CSS

**Objectives**

Students are expected to:

1. Students are able to create static websites using HTML and CSS
2. Students are able to apply the concept of CSS *Display*
3. Students are able to apply the concept of CSS Box Model
4. Students are able to apply the CSS *Flex Box concept*
5. Students are able to apply the concept of CSS *Grid*

**Attention**

This job sheet must be done step-by-step according to the practicum steps that have been given.

**What is CSS?**

CSS stands for "*Cascading Style Sheets*". As the name implies, CSS has the property of "*style sheet language*" which means the programming language used for web design. CSS is a programming language used to design a website page. In designing website pages, CSS uses markers, namely **id** and **class**. CSS can change fonts, font sizes, font colors and formats, set layout sizes, widths, heights and color elements, change the appearance of forms, create responsive website pages and much more.

To design a font can be done by defining the font, to set the color can use color, margins are used to set the distance outside a certain element. Set the font size using "*font size*". The *font*  type uses the "*font-family*" and many others.

**How to Use CSS**

The css file is saved with **the .css** extension. then imported or linked into an HTML or PHP file that we want to design with CSS using the following syntax:

<link rel="stylesheet" type="text/css" href="style.css"/>

The top tag is used to connect the HTML file with the CSS file. Syntax is placed on the html file. The rail and type attributes in the link tag are used to define that what is called or linked is a stylesheet or CSS file, then the href attribute is used to put the location of the CSS file. In the example above, style.css file is located in a folder or a directory with html files. if the CSS file is located outside the folder, it can be associated with:

href=".. /style.css"

If the css file is located in a folder, let's say the folder name is "assets", then to associate it with:

href=".. /assets/style.css"

**Practical Section 1. Connecting HTML with CSS**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a new file in the dasarWeb folder and name it index.html. |
| 2 | Type into the index.html file the code below. |
| 3 | <!DOCTYPE html>  <html lang="en">  <head>      <title>Main Page</title>      <link rel="stylesheet" type="text/css" href="style.css">  </head>  <body>      <h2>Welcome To DasarWeb</h2>  </body>  </html> |
| 4 | Create one new file in the dasarWeb, naming it style.css. Type the code below inside the style.css file. |
| 5 |  |
| 6 | Save the file, then open a browser and run localhost/dasarWeb |
| 7 | In the structure of a web page, index.html will always be the main page of a web. So if there is a index.html in a directory then it will always be called and displayed in the browser. That is why in step 6 it is only typed localhost/dasarWeb page without the file name. |
| 8 | In the code inside the index.html there is a <link> tag in the head where the href attribute refers to the style.css file. |
| 9 | style.css contains code to set the display inside a page that refers to a style.css file, in this practicum it is index.html. So that the display inside the index.html will match the properties set in the style.css. |

**Section using <div> tags**

The <div> tag defines a section in an HTML document. The <div> element is often used as a container for other HTML elements to add *styles* with CSS or to display specific tasks using JavaScript.

**Practical Section 2: Use of div**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a new file named div.html inside the dasarWeb folder. Type the code in step 2 inside the div.html |
| 2 |  |
| 3 | Save the file, then open your browser and run localhost/dasarWeb/div.html |
| 4 | What do you understand from using div on the file? Record below your understanding. (Question No. 1)  The use of the <div> element in the code above serves as a container to group a heading and paragraph together, creating a single display that is easy to manage, for example by applying a light blue background color style. In addition, <div> helps with the layout of web pages, such as wrapping content. |

**Getting to know classes and ids in HTML**

Class and id are used as markers in html, what is meant by markers here is that html elements can be marked with class or id. Elements in HTML are tagged so that they can be manipulated using CSS or JavaScript. The simple shadow is if you have five boxes, all the boxes you have are blue, then you want to change the color of the third box, now this is where the use of class and id comes in, to give a sign or name to your box so that it can be changed, and the other boxes will not change.

The difference between class and id is that the class is called on css or javascript using a "period".", and the id is called on CSS or javascript with the hashtag "#". The advantages of class is the class can be given to many HTML elements and can be called at once, while ID can only work on one tag, meaning one ID name can only be given to one element.

**Practical Section 3: Getting to know classes and ids in HTML**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Complete the code inside the index.html with the following code. |
| 2 |  |
| 3 | Complete the code inside the style.css file so that it becomes the code in step 4 |
| 4 |  |
| 5 | Save both files, then open a browser and run /refresh localhost/dasarWeb |
| 6 | What do you understand from the use of class and id in index.html? Record below your understanding. (Question No. 2)  The use of class and id in index.html is mainly for identifying and styling elements. A class is used to group multiple elements that share the same style, allowing to be applied to many elements at once. An id, on the other hand, is unique and should only be used for one element on the page, making it useful for applying specific styles. So, classes are for reusable styling across elements, while ids are for uniquely identifying a single element. |

**How to Write CSS**

CSS uses selectors (id and class) to determine the elements that will be modified by CSS, if it is likened to HTML as a pillar on a house building, then CSS functions as paint and decoration on the house building. There are three CSS writing method techniques, namely:

* *Inline CSS Style*

It is CSS created in an HTML tag that only applies to the document it is enclosed in. Usually this technique is used for special formatting of an HTML element and is not used to format the entire element in a web document.

Example:

<h1 style="color:blue; margin-left:30px;" >This is a heading.</h1>

* *Internal CSS Style*

An internal style sheet for a web page applies only to that web page. The internal style sheet is defined in the HEAD section of an HTML page, in the <style> tag as follows:

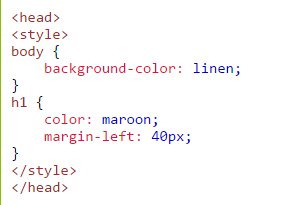


Figure 1. Internal CSS

* *External CSS Style*

It is ideal for use on the web with many pages. By using the External Style Sheet, the appearance of the entire website content can be changed by changing just one file. *External Style Sheets* must not contain html tags, and are stored in a file with the \*.css extension

Example:



Figure 2. External CSS

The way to call an *External Style Sheet* in a web page is by using the <link> tag with the rail attribute placed on the *section head*. Example:

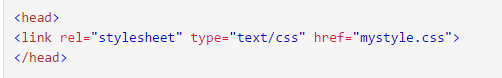


Figure 3. Using an external CSS file into HTML

**Practical Section 4: Changing the *Background* of a Web Page with CSS**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Add the code snippet in step 2 below to the style.css file |
| 2 |  |
| 3 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 4 | Record your observations and write your answer below. (Question No. 3)  From the result shown in the browser, I observe that the CSS styling in the body tag sets the page background color to lightcyan and the default text color to white. This shows that the body style provides a general background and text color. |

**Practical Section 5: Using *images* for *backgrounds***

The background-image *property assigns* one or more *background images* to an element. By default, the background image is placed in the top left corner of the element, and it repeats vertically and horizontally.

Tip: The background of an element is the total size of the element, including *padding* and *borders* (but not margins).

Tip: Always set the background color to use if the image is not available.

Practicum Steps:

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | In the style.css file change the value from background property to bunga2.jpg (or the name of image file as background) as in the code snippet in step 2. |
| 2 |  |
| 3 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 4 | Record your observations and write your answer below. (Question No. 4)  My observation is that when using a background color, the browser simply fills the entire page with a solid color (lightcyan) making it uniform and flat. In contrast, when using a background image with url(), the page background is filled with the chosen image (bunga2.jpeg), which gives a more decorative appearance depending on the image. The main difference is that background color provides a plain and simple look, while background image adds visual variety, but it also depends on the correct file path so the image can be displayed. |
| 5 | Add an image file named field1.jpg and tree1.jpg (or the name of images file as background and object) to the img folder. |
| 6 | Type the following code on style.css. |
| 7 |  |
| 8 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 9 | Your code runs fine if it looks like the one below: |
| 10 | Record your observations and write your answer below. (Question No. 5)  My observation is that the CSS code applies two background images to the body, the first image (tree1.jpg) is set with no-repeat, so it appears only once on the left side of the page, while the second image (field1.jpg) is set with repeat, so it tiles across the remaining background. As a result, the tree image is fixed as a single decoration, and the field image repeats to cover the rest of the page and creating a layered background effect. |

**Practical Section 6: Margins and Padding on CSS**

**Getting to Know Margins in CSS**

The margin is the outer side of an element. for example you want to set the distance between the elements. You can use *the margin syntax* to set it. There are several outer sides of margin which are,

* The top margin is written in the CSS with a 'margin-top',
* the bottom margin or the outer spacing of the bottom is written in CSS with 'margin-bottom',
* 'margin-left' as the outer spacing to the left of the element, and
* 'margin-right' is the outer side on the right side

But if you only use the 'margin' syntax it will automatically set the top, bottom, left and right spacing of the element.

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Rename index.html to backgroundCSS.html |
| 2 | Create a new file inside the dasarWeb folder named index.html. This means that you have a new index.html file. |
| 3 | Type the following code on index.html |
| 4 |  |
| 5 | Create a new file named styleMargin.css in the dasarWeb folder. |
| 6 | Type the following code toto the styleMargin.css |
| 7 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 8 | Note here what you observe from the code above. (Question No. 6)  My observation from the code and its browser output is that two <div> elements are styled differently using CSS classes. The first box (.box) has a blue background, width of 300px, height of 200px, and margin of 30px, so it appears on the left with some spacing from the edges. The second box (.box-dua) has a black background, width of 200px, height of 100px, and is shifted to the right using margin-left: 500px. Both boxes contain <h1> elements, and the h1 style makes the text color pink, so the text inside each box appears in pink against the contrasting background. |
| 9 | Change the margin value in the settings .box to margin: 300px;. Observe what the difference is. |
| 10 | Note here what you observe from the code above. (Question No. 7)  The first blue box will be pushed far down and to the right because the margin applies 300px of spacing on all sides, making the box appear more toward the center of the page instead of near the top-left corner; the second black box remains in its original position with margin-left 500px, so the distance between the two boxes becomes much larger. |

* Getting to Know *Padding* in CSS

Padding is the inner side of an element. We can use syntax padding to set the spacing on the inner sides of an element that we specify. Just like the margin has sides. such as top, left, right, bottom. The type of padding is the top padding written in CSS with 'padding-top' which means setting the inner side of the top of an element, the bottom pad or the spacing in the bottom is written in CSS with 'padding-bottom', 'padding-left' as the inner spacing on the left side of the element, and 'padding-right' is the outer side on the right. If you only use the 'padding' syntax, it will automatically set the top, bottom, left and right spacing of the inner element.

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Rename index.html to marginCSS.html |
| 2 | Create a new file in the dasarWeb folder named index.html. This means that you have a new index.html file. |
| 3 | Type the following code to index.html |
| 4 |  |
| 5 | Create a new file named stylePadding.css in the dasarWeb folder |
| 6 | Type the following code to stylePadding.css |
| 7 |  |
| 8 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 9 | Record your observations and write your answer below. (Question No. 8)  The h1 and h2 headings are centered, with h1 text colored in cornsilk and h2 left in the default text color. The .box element has a blue background, fixed size of 300px by 200px, and a padding of 20px, meaning its content will be spaced inward from all sides, giving it breathing room inside the box. The .box-dua element has a red background, larger dimensions of 600px by 100px, and specifically a left padding of 70px, which shifts its content inward from the left edge while the other sides remain at default padding. The styling shows a clear contrast between the two boxes in terms of size, color, and padding behavior, making the first box more compact and evenly padded while the second one is wider with extra space only on the left. |
| 10 | Change the padding value in the .box setting to padding=200px; |
| 11 | Record your observations and write your answer below. (Question No. 9)    Changing padding to 200px makes the blue box very big and pushes its content away from the edges, creating a huge empty space around the text inside. |

**Practical Section 7: Font Settings in CSS**

Some of the css syntax used to set the font:

* *font-size* is used to set the font size
* *font-weight* is used to adjust the thickness of the font
* *font-family* to change the font type
* *font-style* is used to change the style of the font.
* *color* is used to change the color of the font

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Rename index.html to paddingCSS.html |
| 2 | Create a new file in the dasarWeb folder named index.html. This means that you have a new index.html file. |
| 3 | Type the following code to index.html |
| 4 |  |
| 5 | Create a new file named styleFont.css in the dasarWeb folder |
| 6 | Type the following code to the styleFont.css |
| 7 |  |
| 8 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 9 | Record your observations and write your answer below (Question No. 10)  The .tulisan\_satu class makes the text red with a sans-serif font and a normal style, giving it a clean and strong look. The .tulisan\_dua class makes the text green, larger in size (24pt), italicized, which gives it an emphasized and stylish appearance while the .tulisan\_tiga class turns the text blue, bold, and oblique, resulting in a strong and slightly slanted effect. |

**Practical Section 8: Setting *Up Hyperlinks* with CSS**

*Hyperlinks* are links that are created to redirect pages when clicked. *Hyperlinks* or links are created using tags and ending with tags in HTML. There are 4 states that are owned by html *hyperlinks* and can be manipulated using css. Namely:

* *link*. is an ordinary active link.
* *visited*. is the status of a link that has been visited.
* *hover*. is the status of a link when the mouse cursor is placed on it.
* *active*. is the status of a link or *hyperlink* when it has been clicked.

*The syntax* is as follows:

* *a:link* = for regular links
* *a:visited* = is the status of a link that has been visited.
* *a:hover* = is the state of a link when the mouse cursor is placed on it.
* *a:active* = is the status of a link or hyperlink when it has been clicked.

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Rename index.html to fontCSS.html |
| 2 | Create a new file in the dasarWeb folder named index.html. This means that you have a new index.html file. |
| 3 | Type the following code to index.html |
| 4 |  |
| 5 | Create a new file named styleLink.css in the dasarWeb folder |
| 6 | Type the following code to styleLink.css |
| 7 |  |
| 8 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 9 | Record your observations and write your answer below (Question No. 11)  The hyperlink is styled using CSS to appear with a font size of 20pt and by default shows as blue because of the link selector. When the user hovers over the link, its color changes to red due to the :hover rule. The link opens in a new tab (target="\_blank") when clicked, and the styling makes it more noticeable and visually engaging compared to the browser’s default hyperlink appearance. |
| 10 | Add code to styleLink.css to change the color of the link to greenyellow after the link is visited. |
| 11 | Write the code you added here (question no 12) |

**Practical Section 9: Formatting Text with CSS**

The text formatting settings in CSS are as follows:

1. *color*: to set the color of the text, the value can be filled in as a color or color code
2. *text-align:* to set the align position on the text or flat text, the values that can be filled in include center to make the text center-aligned, left to make the text left-aligned, right to make the text right-aligned and justify to make the text right-aligned and left-aligned.
3. *text-decoration:* to set the decoration of the text, the value is none to make the text have no decoration, overline to make the text have a line at the top of the text, line-through to create a scrawly line on the text, and underline to make a line at the bottom of the text (underline).
4. *text-transform*: to set capital letters in text, values that can be used include uppercase to make text uppercase, lowercase to make text lowercase, and capitalize to make the initial letter of each word uppercase.
5. *text-indent:* to set the spacing of paragraphs in the text, the value that can be used is in the form of pixel values and others as needed.
6. *letter-spacing*: to adjust the distance between characters in text, the value filled in the form of pixel values and others.
7. *Word-spacing:* To set the spacing between words in the text, the value filled in is also a pixel value.
8. *line-height*: to set the distance between lines in the text value that is filled in the form of a value.
9. *text-shadow*: to set the shadow effect on the text, the first filled value fills in the value for the left and right distances, and the second fills the top and bottom spacing and the third fills the color. For the example of writing it is 2px 5px blue.
10. *vertical-align*: to set the align in a vertical form in the text value used is left to make the text aligned left, right to top and center to center.

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Rename index.html to linkCSS.html |
| 2 | Create a new file in the dasarWeb folder named index.html. This means that you have a new index.html file. |
| 3 | Type the following code to index.html |
| 4 |  |
| 5 | Create a new file named styleText.css in the dasarWeb folder |
| 6 | Type the code in step 7 below to the styleText.css . |
| 7 |  |
| 8 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 9 | Record your observations and write your answer below (Question No. 13)  The observation is that the first line of text "Mengatur format text dengan CSS" appears in blue color because it is styled with the CSS class .tulisan\_warna, while the following long paragraph uses the default black color since the class par1 does not have any CSS styling applied. |
| 10 | Add code to styleText.css to change the position of the text to center by adding text-align:center; and add decorations in the form of underlines. |
| 11 | Write the code you added here (question no 14) |
| 12 | Add code to styleText.css to add an underline to the text. |
| 13 | Write the code you added here (question no. 15) |
| 14 | Add the code to styleText.css to give the spacing between characters in the existing paragraph to 5px with letter-spacing. |
| 15 | Write the code you added here (question no 16) |

**Practical Section 10: Getting to Know CSS Positions**

Position in CSS is used to set the position of an HTML element. This CSS position property is used to determine the position of an HTML element as desired.

In general, to create a position or set the position of an element we must use other css properties such as setting top, left, bottom, right to set the position of an element, but these properties will not work if the position has not been set first, this is because other properties depend on the position that is set.

Some CSS properties that can be used to position an HTML element are:

* *Static*: Position static is used to set an element to static by default. Elements will follow the normal position by default, elements are not affected by properties such as top, bottom, left and right.
* *Relative*: An HTML element that uses relative position will be located at the normal position. Setting the top, right, bottom, and left properties of an element positioned "relatively" will make it far from its normal position. Other content will not be adjusted to fit the gaps left by those elements.
* *Fixed*: An HTML element set to a fixed position will have a fixed property without any changes even if the website page is scrolled. Left, bottom, top and right settings apply to fixed positions.
* *Absolute*: HTML elements that use the absolute position will be positioned relative to the other elements that precede it closest to it, not relative to the screen normally.
* *Sticky*: An element with a position: sticky; positioned based on the user's scroll position. Sticky elements alternate between relative and fixed, depending on the scroll position. It is positioned relative until a certain offset position meets in the viewport - then "sticks" in place (such as position: fixed).

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Rename index.html to textCSS.html |
| 2 | Create a new file in the dasarWeb folder named index.html. This means that you have a new index.html file. |
| 3 | Type the following code to index.html |
| 4 |  |
| 5 | Create a new file named stylePosition.css in the dasarWeb folder |
| 6 | Type the following code to stylePosition.css |
| 7 |  |
| 8 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 9 | Record your observations and write your answer below (Question No. 17)  The static position is the default, where elements follow the normal document flow and cannot be adjusted with top, left, right, or bottom properties. The relative position allows an element to remain in the normal flow but can be shifted from its original location without affecting the space it originally occupied. The fixed position, on the other hand, allows an element to remain in the same spot relative to the browser window even when the page is scrolled, making it useful for navigation bars or sticky footers. The absolute position places an element relative to its nearest ancestor that has a positioned property (anything other than static), otherwise it is placed relative to the document body. Finally, the sticky position acts as a hybrid between relative and fixed, it behaves like a relative element until the page is scrolled to a certain threshold, at which point it sticks to the defined position like a fixed element. |

**Practical Section 11: Using Floats**

The Floating technique in the web design part is the most needed need. The float property is used to position and format content, for example. Make the image hover to the left of the text in the container. One of the most common examples of the use of floats is when we want to create a website postigan that is located on the side of the content text. A float property can have any of the following values:

* left - Floating element to the left of its container
* right- Floating element to the right of its container
* none - The element is not floating (will be displayed exactly where the text appears). It's standard
* *inherit* - This element inherits the float value from its parent

In its simplest use, the float property can be used to wrap text around an image.

Practicum Steps:

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Rename index.html to positionCSS.html |
| 2 | Create a new file in the dasarWeb folder named index.html. This means that you have a new index.html file. |
| 3 | Type the following code to index.html |
| 4 |  |
| 5 | Create a new file named styleFloat.css in the dasarWeb folder |
| 6 | Type the following code to styleFloat.css |
| 7 |  |
| 8 | Save the file, then open the browser and run /refresh localhost/dasarWeb |
| 9 | Record your observations and write your answer below. (Question No. 18)  The <h2> element is styled to appear centered with a blue-violet color, while the image is given a fixed width of 100px, floated to the left, and spaced with a right margin so that the paragraph text flows neatly on its right side. This creates a clean layout where the image and text appear aligned in a visually appealing way. |

**What is CSS Layouting?**

CSS *layouting* is a technique for arranging the layout of a web page using CSS code. The purpose of CSS Layouting is to make the page look neat and attractive as expected by the creator. CSS *layouting* consists of several parts namely; dimensions, *overflow*, Box model, float and Position

**CSS *Display***

HTML tags are used to give a 'meaning' to a piece of content (e.g. p for paragraph, h1 for main *heading* and so on). The <div> and <span> tags have no meaning, they are both used to group HTML tags and provide Description to them.

**Practical Section 12. CSS *Layouting***

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a new file in the dasarWeb folder and name it csslayouting.html |
| 2 | Type the following code to csslayouting.html |
| 3 |  |
| 4 | Save the file, then open the browser and run /refresh localhost/dasarWeb/csslayouting.html |
| 5 | Then create a <div> on the same html page as in the code below |
| 6 | A screenshot of a computer  Description automatically generated |
| 7 | Observe whether the results of the two programs are the same or different, explain the reason (Question No. 19)  The display in the browser look exactly the same. Although the second code adds <div> elements with classes such as header, navigation, main, and copyright, the <div> element itself is just a container. The <div> tag functions as a container. Without a CSS file that calls classes such as .header, .navigation, .main, and .copyright, <div> will not change the layout or visual appearance in any way. Without CSS code targeting those classes to set styles such as color, size, or position, the <div> element has no visual effect on the display. Therefore, both HTML codes will produce identical layouts and displays in the browser. |
| 8 | Add *styles* to the *navigation* class and *the main* **class** as in the code below |
| 9 |  |
| 10 | Run the code andcapture the results. Explain what happened (Question No. 20)    The code is applying CSS to define the background color for specific HTML elements. Inside the <head> section, the <style> tag contains rules that target elements based on their class names. Specifically, the .navigasi class is being styled with a pink background color, while the .main class is being styled with a light green background color. |

**Value of *display***

* Inline display

HTML elements that by *default* do not add new lines when created. The characteristics of *inline dispay* are

1. The width and height of the elements are in accordance with the content in them
2. Cannot adjust the height and width of inline elements
3. *Margins* and *padding* only affect elements horizontally, not vertically

The *inline*  elements are as follows; **b, strong, i, em, a, span, sub, sub, button, input, label, select, textarea**

**Practical Section 13. Inline Display**

|  |  |  |
| --- | --- | --- |
| **Step** | | **Description** |
| 1 | Delete the *style* in **step 9** **Practical Section 12**. Add 5 links to the navigation *class* as shown in the code below. | |
| 2 | A screenshot of a computer  Description automatically generated | |
| 3 | Run the code andcapture the results. Explain what happened (Question No. 21)    The display on the browser will look the same as before, where the color on certain parts of the text is missing because the CSS code has been deleted. But here, there is an additional link 5 that appears after link 4. | |

* *Display inline-block*

If in an *inline* element we can't set the height and width of an element, then we can set it using *inline-block*. Element has no property *by default***.**

**Practicum Section 14. Display *inline-block***

|  |  |  |
| --- | --- | --- |
| **Step** | | **Description** |
| 1 | Add *weight*, *height* and *display* to the *style* in element **a** as shown in the code below | |
| 2 | A screen shot of a computer  Description automatically generated | |
| 3 | Run the code andcapture the results. Explain what happened. (Question No. 22)    The CSS code for the <a> (link) element makes the link background pink, gives it a width and height of 200px, and changes its behavior from inline to inline-block. With display: inline-block, the link will now function like a block element that can have dimensions, while remaining in the same text flow as inline elements. | |

* *Display Block*

*Block* is an HTML element that by *default* adds a new line When it is created If it is not set in width, then the *default*  width of the *block*  element will meet the width of  *its browser*/*parent* so that we can set the height and width of the *block element.* Inside the *block* element, we can store tags with *inline elements*, *inline-blocks*, or even *block* elements again. Examples of *block*  elements are; h1-6, p, ol, ul, li, form, hr, div

**Practicum Section 15. Display *block***

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Style the h1, h2 and p elements with a light-green *background color* as shown in the code below |
| 2 | A screenshot of a computer program  Description automatically generated |
| 3 | Run the program and try to resize the web page. *Capture* the results, observe and explain what happened. (Question No. 23)    All links (a) or all link elements (<a>) on the web page will have a pink background color. All titles and paragraphs (h1, h2, p), all main title elements (<h1>), secondary titles (<h2>), and paragraphs (<p>) will have a light green background color. |
| 4 | Then add a style to the main class as in the code below |
| 5 | A screenshot of a computer program  Description automatically generated |
| 6 | Run the program. *Capture* the results, observe and explain what happened. (Question No. 24)    The web page's appearance changes in three specific ways. First, all anchor tags <a> get a pink background color. Second, any element with the class main will be given a fixed width of 600px and a light green background. Third, and most specifically, any <h2> heading that is inside an element with the class main also have a pink background color. |

* *Display none*

The last display value is *none*, *none* can be used to remove an element

**Dimensions and *Overflow* on CSS**

Dimensions have two *properties* in CSS, namely *width* for width and *height* for height. Units of dimensions are various including **px, %, in, cm, mm pc pc**.

*Overflow* is a CSS *property* that is used to set the behavior of an element that is not enough on a *parent.* There are four *values* of *property overflow*:

1. *Visible*: *Value default*
2. *Auto*: CSS will automatically add *scrolls* if there isn't enough content
3. *Hidden*: the content will be hidden or invisible
4. *Scroll*: like auto, will bring up *a scroll*, but if enough *content* scrolls will still exist

***Box model* on CSS**

Every element on a *website*  page is in a *box*. We can set the size and position of the box. We can give a color/image as the *background* of the box. The box model in CSS defines the 'box' generated by an element, and then displays it according to its visual format. The CSS box model consists of 4 components namely; margins, borders, padding and *content* as shown in the image below.

A screenshot of a computer

Description automatically generated

Figure 4. Box Model Components

1. *Margin*: the transparent area around the box (outside the *border*)
2. *Border*: the border around *the content* and *padding*
3. *Padding*: a transparent area inside the box (between *the content* and *the border*)
4. *Content*: the actual content in the *box*, it can be text or images

Table 1. How to set up a model box property

|  |  |  |
| --- | --- | --- |
| ***Margin*** | ***Padding*** | ***Border*** |
| *Margin-top*  *Margin-right*  *Margin-bottom*  *Margin-left*  *Margin* | *Padding -top*  *Padding -right*  *Padding -bottom*  *Padding -left*  *Padding* | *Border -top*  *Border -right*  *Border -bottom*  *Border -left*  *Border* |

* *Box Model: Margin*
* *Ovelapping* margin occurs when we combine two margins. Left and right or up with bottom. In theory, if it happens, the greatest value will be taken.
* *A negative* margin will make the box go in the opposite direction. It can be used if we want to hide elements.
* *Auto* margin is the value that we can assign to the special margin for the left margin and the right margin. This auto will make the element in the middle of the browser's web page
* *Shorthand* margin is a way of abbreviating margin writing.

**Practicum Section 16. *Box Model: Margin***

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create 2 files as shown in the image below. The first file with the name margin.html and the second file marginstyle.css |
| 2 | A screenshot of a computer program  Description automatically generated |
| 3 | Run the program, observe the results and explain it (Question No. 25)    The .satu div is the largest with a size of 200px and a light green background, the .dua div is a medium-sized box at 100px with a light blue background, and the .tiga div is the smallest at 50px with a violet background. Because no CSS properties like margin or float were applied, they all maintain their natural flow and are positioned directly below each other, each starting on a new line. |
| 4 | Next is to give *a margin*, where *the margin* is the *transparent area* around the box. Add *a style* to the **margin-top**  marginstyle.css of 100px on class **.satu**, save it and then run it in a web browser. Capture and observe the results (Question No. 26)      The light green box with the number “1” is positioned 100px from the top edge of the browser. Margins are transparent spaces outside the borders of elements, so they do not have a background color. |
| 5 | Then add another ***margin*** size as in the following code |
| 6 |  |
| 7 | Run the program. *Capture* the results, observe and explain what happened. (Question No. 27)    Elements with the .satu class will be a 200x200px box with a light green background. This box will have transparent space around it called margin. Specifically, margin-top: 30px will give it a 30px space from the top, margin-left: 100px will move it 100px to the right from the left edge, and margin-bottom: 150px will create a large space, namely 150px, between this box and the .two element below it. Meanwhile, the .dua and .tiga elements will still appear as boxes with their respective sizes and colors, but their positions will be affected by the margin applied to the .satu element. |

* *Box Model: Padding, Border & Box Sizing*

*Padding*

How to use *padding* is the same as the margin, namely, it cannot be used *negative*, it cannot be given *an auto* value and affects the size of the *box* of an element.

*Border*

How to write it;

*Border: width style color*;

*Style* on border; solid, dotted, dashed, double

*Box sizing*

*Box sizing* is a property that accepts the padding and *border* values on an element including the total value of the *width* and *height*  of an element.

*box-sizing*: *content-box* (*default*)| *border-box* | *Unser* | *initial* | *inherit*;

***Flex Box***

It is a 1-dimensional layout model that can adjust the distance and alignment between items in a *container*. What is meant by one-dimensional is that it can only set one dimension at a given moment, between rows or columns, it cannot be both at the same time.

*Flex Box* or Flexbox *Layout Module* is a module that offers an effective way to arrange, align and distribute the spacing between items in a *container*, even though the size is dynamic or we don't even know.

A diagram of a blue rectangular object with white text

Description automatically generated

Figure 5. Terms on flex Box

(source; <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>)

* ***Main axis***; The main axis of a *container* that determines the order of the horizontal placement of items
* ***Main start****/****main end***; Start and end of items stored in containers
* ***Main size***; size (*width/height*) of the *container* which will make the dimensions of the items relative to the size

***Properties* on *containers***

*A container* is a wrapper of an element

A cartoon of a purple rectangular object

Description automatically generated

Figure 6. Container

***Display***

A blue background with white text

Description automatically generated

*Display* Makes a *parent*  element a flex box, and makes the element within it flex as well.

***Flex- direcrtion***

*Flex-direction* Sets the direction/order of the items in the *container*

A diagram of orange squares

Description automatically generated with medium confidence

Figure 7. Flex- direction

A screen shot of a computer

Description automatically generated

***Flex-wrap***

By *default*, all *items* in the *container* will be in one row even if the size is no longer enough, wrap allows you to move items to the bottom row

A purple and orange rectangular object with white dotted line

Description automatically generated

Figure 8. Flex-Wrap

A blue background with yellow and orange text

Description automatically generated

For other properties, you can visit the website https://css-tricks.com/snippets/css/a-guide-to-flexbox/

**Practicum Section 17. *Flex Box***

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Type the code below then save the file with the name flexbox.html |
| 2 | A screenshot of a computer screen  Description automatically generated  A screenshot of a computer  Description automatically generated |
| 3 | And for the style1.css file as follows |
| 4 | A computer screen shot of a black screen  Description automatically generated |
| 5 | Run the program, observe the results and explain it (Question No. 28)    The webpage displays a single-column layout where the main content and sidebars are stacked vertically. This is the default behavior for block-level elements like div, h2, h3, and p. The CSS adds a light gray background to the entire page, removes the default margin and padding from the HTML and body elements, and sets a basic font style and line height for readability. |
| 6 | Add *a style* to the **container-satu** class on the style1.css file as in the code below |
| 7 | A screen shot of a computer program  Description automatically generated |
| 8 | Run the program. *Capture* the results, observe and explain what happened.  (Question No. 29)    Width: 800px; sets the width of the content box to 800 pixels, while margin: 50px auto; serves to position the box horizontally in the center of the page and give it a 50-pixel margin from the top and bottom. Additionally, background-color: #fff; makes the box background white, and padding: 20px; adds empty space inside the box, between the content and the box edges. Finally, box-sizing: border-box; ensures that the total width of the box, including padding, will not exceed 800 pixels. All of these properties work together to create a neat, centered content box that is spaced away from the edges. |
| 9 | Add a display property on a container-satu selector with a flex value. Capture and explain the result (Question No. 30)    \  When we add display: flex; to .container-one, that element becomes a flex container. This means that all direct child elements within it (.main-column, .sidebar-one, and .sidebar-two) automatically become flex items. The most noticeable change is that these elements, which were previously arranged vertically (because divs are block elements), will now be arranged horizontally in a single row. Each flex item will try to occupy space side by side. |

**CSS *Grid***

*The CSS grid layout module* offers a grid-based *layout system* with rows and columns that makes it easier to design web pages without using floats and positions. It is a new CSS module to define a grid-shaped layout system in 2 dimensions (rows and columns)

A screenshot of a computer

Description automatically generated

Figure 9 Grid Layout Module

*Grid element*

A grid layout consists of a single parent element with one or more child elements

*CSS Grid Terminology*

* *Grid container:* The element that wraps the grid, defined by writing: display: grid;
* *Grid items*; Element element located (1 level) in the container grid
* *Grid line*; Horizontal (column) or vertical (row) lines that separate a grid into sections and are marked with numbers
* *Grid cell*; Intersections/confluences between rows and columns in a grid
* *Grid area*; A collection of more than one grid of cells that form a box
* *Grid track*; Size/distance between 2 grid lines, can be horizontal (column) or vertical (row)
* *Folding grid*; Distance between grid tracks/cells

For more details about the grid, you can learn on the following website <https://www.w3schools.com/css/css_grid.asp>

**Practicum Section 18. *CSS Grid***

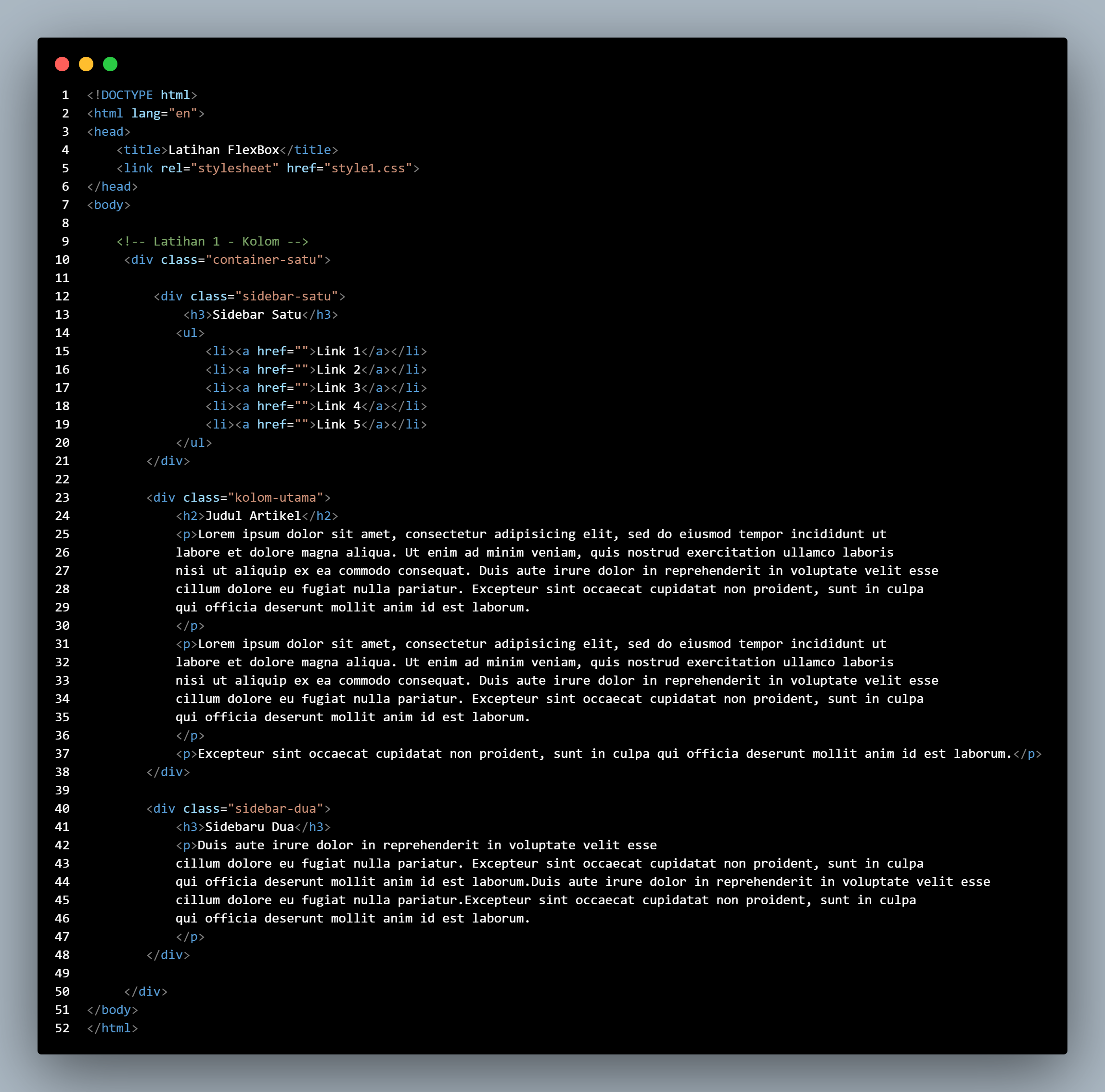
|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Type the code below then named with grid.html |
| 2 | A screen shot of a computer  Description automatically generated    A screen shot of a computer  Description automatically generated |
| 5 | Run the program, observe the results and explain it. (Question No. 31)    The .container is the grid parent, and the header, aside, nav, main, and footer elements are its children, or grid items. The grid-template-areas property is the key to this layout, defining a 4x2 grid with named areas. It places the header and footer elements to span the full width of the grid, while the aside, nav, and main elements are positioned in the middle rows. The grid-template-columns and grid-template-rows properties further define the size of these areas: the columns have a proportional width ratio of 1:1.5 (1fr 1.5fr), and the rows have a proportional height ratio of 1:1:1.3 (1fr 1fr 1.3fr). This combination of properties results in the complex, two-dimensional layout seen in the image, where elements are neatly arranged into columns and rows without the need for floats or absolute positioning. |

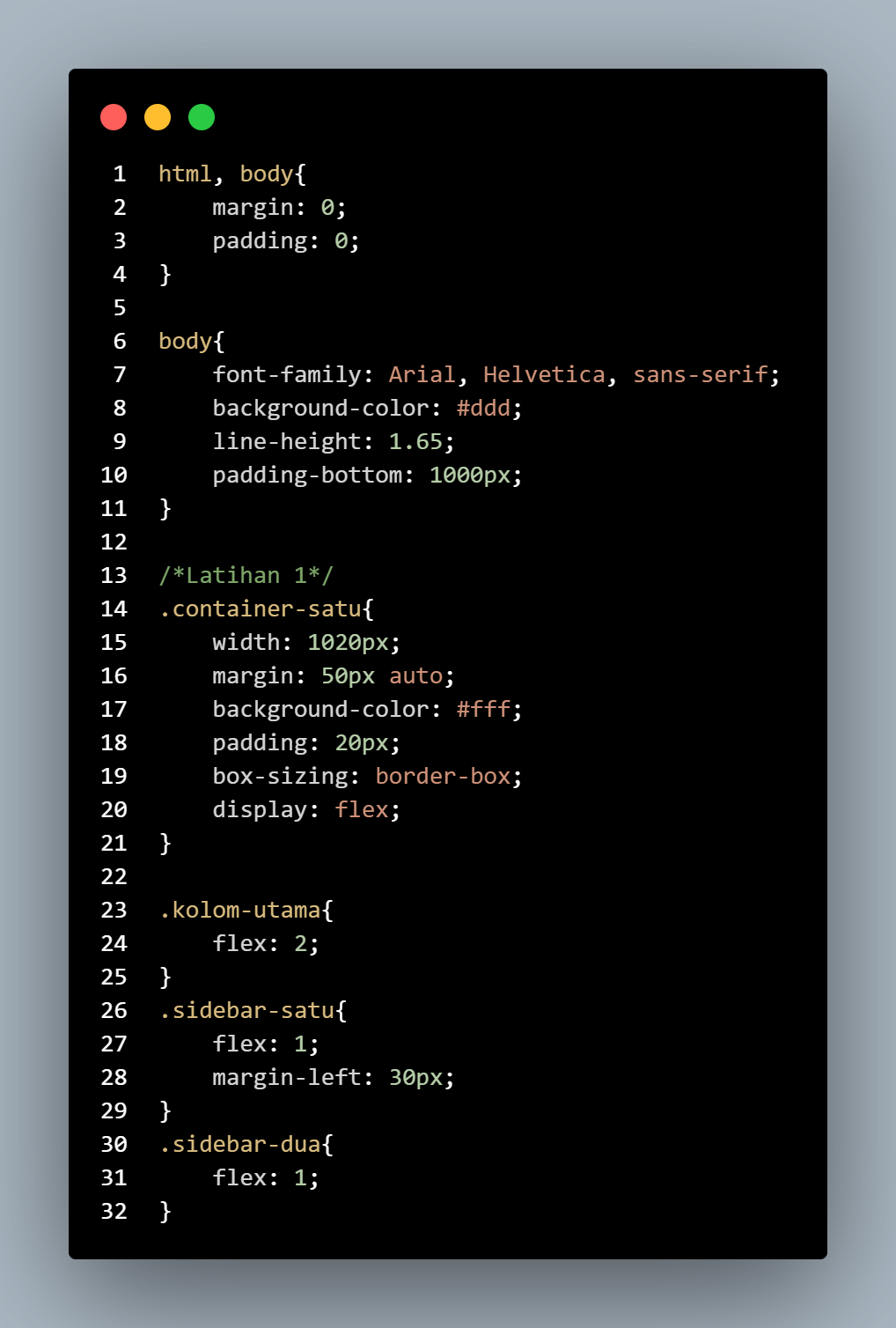
**JOBSHEET 2 TASKS** (Question No. 32)

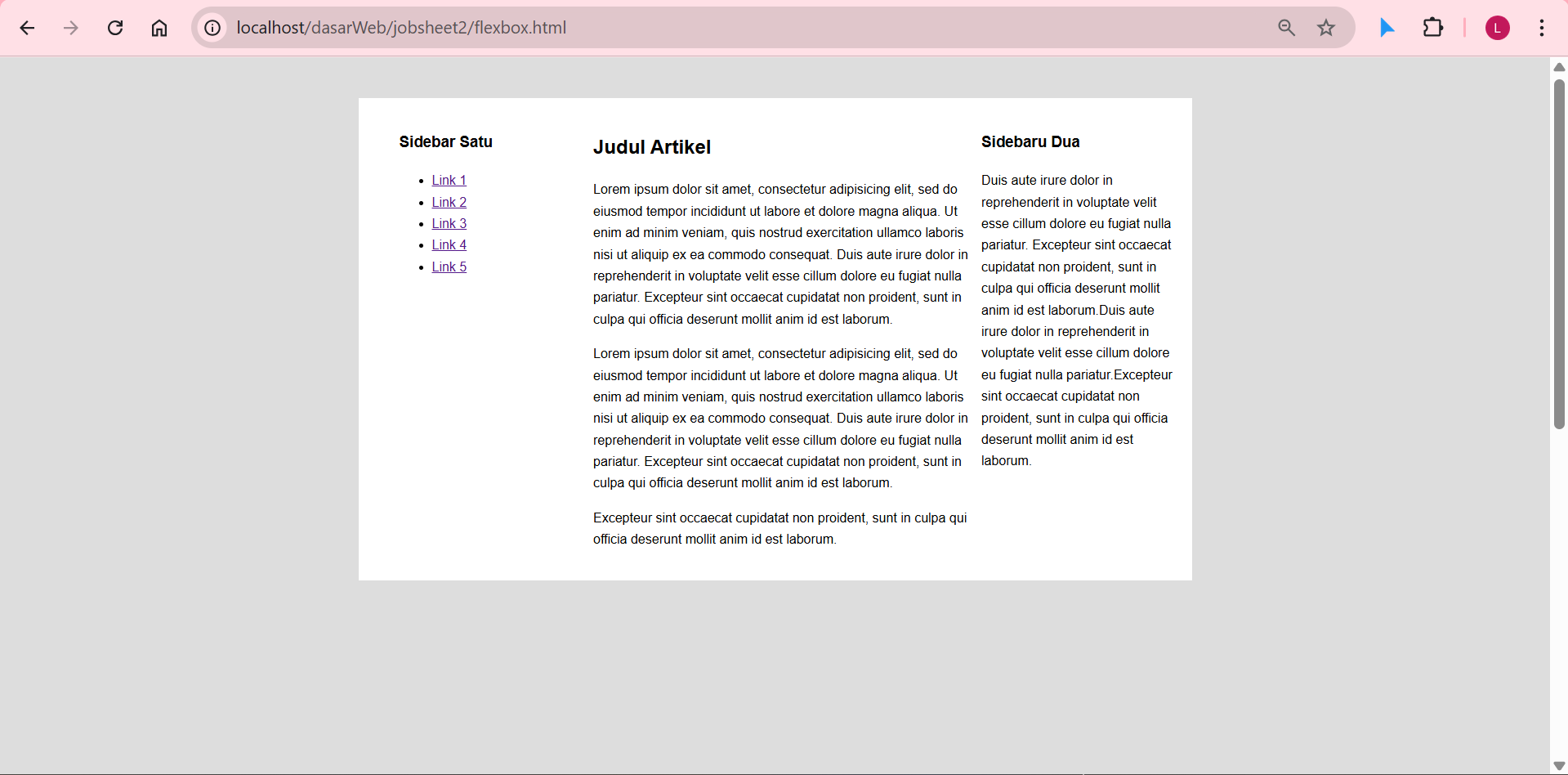
1. Make the previous *flex-box*  result look like this

A screenshot of a computer

Description automatically generated

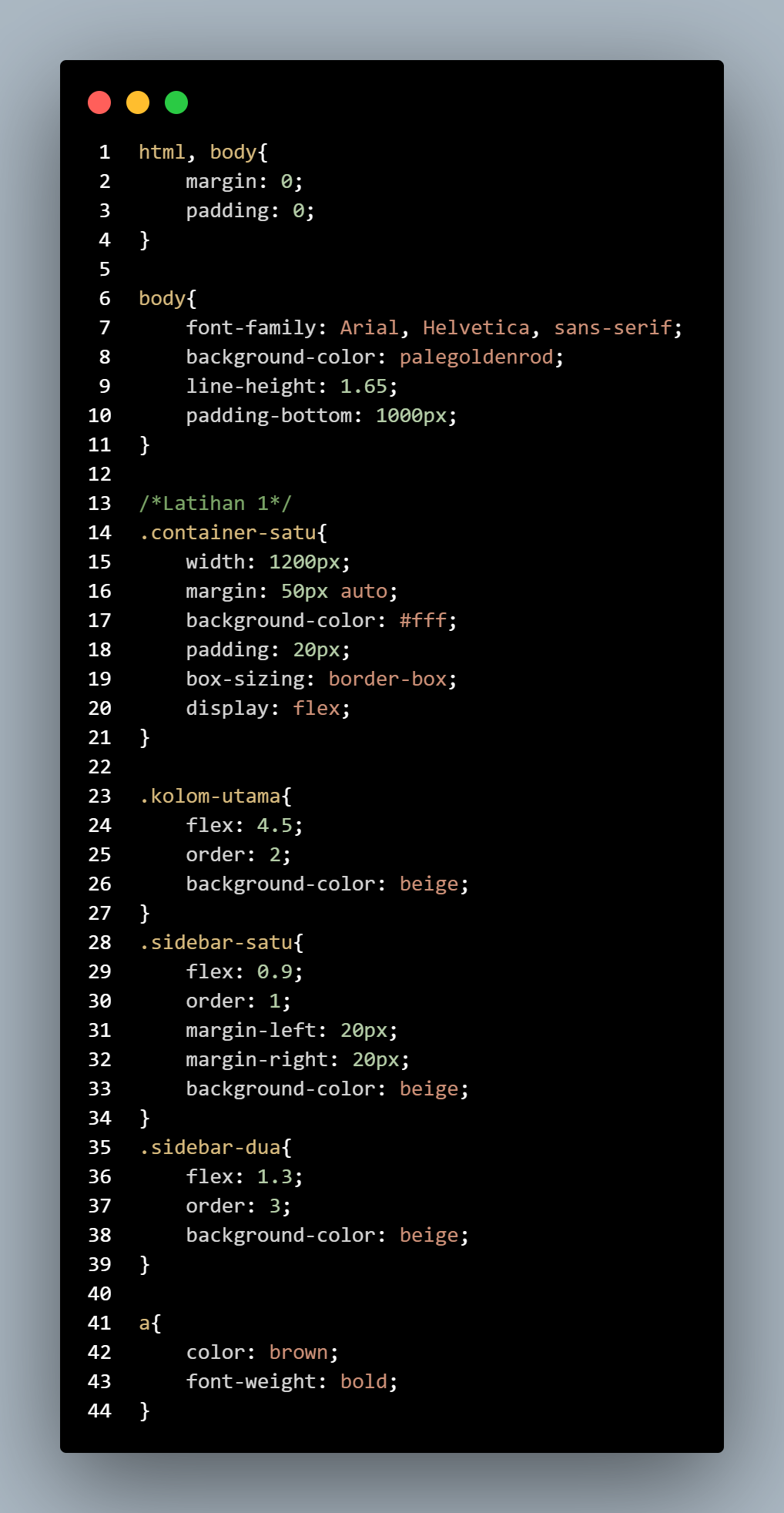
Code :  




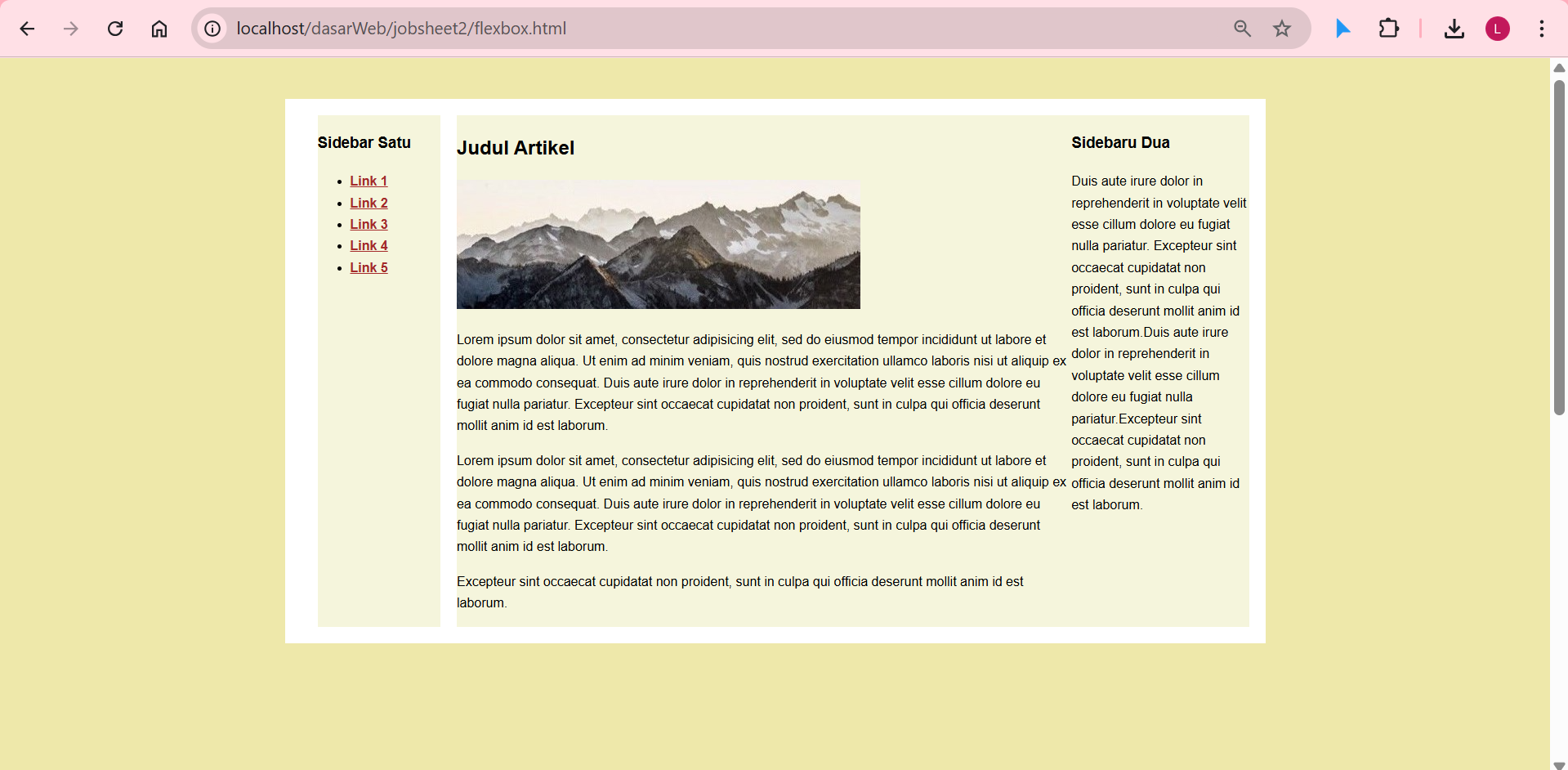


1. Add images and apply color play to the background to make it more interesting.

**Keywords**; Use *the Flex* and *Order* properties on each  *of its item selectors*. Then *capture* the code and the result







Reference:

1. Jason Beaird, The principles of Beautiful Web Design
2. Rian Ariona, Learn HTML and CSS (Fundamental Tutorial on learning HTML and CSS)
3. Adi Hadisaputra, HTML and CSS Fundamentals from the Roots to the Leaves of John Duckett, HTML and CSS design and build websites
4. <https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements>
5. <https://css-tricks.com/almanac/properties/d/display>
6. <http://www.w3.org/TR/CSS2/box.html>)
7. <http://www.w3schools.com/css/css_boxmodel.asp>)

Github link : <https://github.com/Lovie-Tonimba/semester3-PemrogramanWeb.git>