**Objective:** This worksheet will guide you through the process of learning and practicing Grid Layout in web development. Follow the instructions and use the provided space to complete the exercises.

**Part 1: Understanding Grid Layout Basics**

1. **Grid Container and Grid Items**:
   * Define what a grid container and grid items are in the context of Grid Layout.

A grid container is an HTML element that is the parent for a grid layout of divs. It defines the boundaries for the grid and is responsible for the formatting of the grid items.

* Provide an example of an HTML structure that represents a grid container and its grid items.

initial-scale-I 
6 
20 
30 
< !DOCTYPE html) 
<html 
< head) 
(meta 
(meta name. "viewport" 
<tit1e>Document</tit1e> 
• grid- container { 
display: grid; 
grid-template-columns 
gap: lepx; 
align-items: center; 
. grid-container div { 
text-align: center; 
border: 2px solid Oblack; 
< / style) 
< / head > 
< body 
<div 
(div 
(div 
(div 
(div 
< / body > 
< /html> 

1. **Grid Lines and Grid Cells**:
   * Explain what grid lines are and how they function in Grid Layout.

Grid lines are imaginary horizontal and vertical lines on the html page that form a grid within the container. The lines are numbered starting from one and increase sequentially in both rows and columns. These lines can be used as reference points to place the other objects inside the grid.

* Describe what a grid cell is and how it relates to grid lines.

A grid cell is a rectangle that is created by four grid lines. They are used to position and size the different grid items.

* Draw a simple diagram to illustrate a grid container with grid lines and cells.

**Part 2: Exploring Grid Layout Properties**

1. **Common Grid Properties**:
   * List and briefly explain at least five common CSS properties used in Grid Layout, such as **grid-template-columns**, **grid-template-rows**, **grid-gap**, etc.

**Grid-template-columns:** defines the size and number of columns in the grid container.

**Grid-template-rows**: defines the size and number of rows in the grid container.

**Grid-gap**: the size of the space between grid items.

**Grid-template-areas**: Each grid item is associated with a named area, and you define the placement of items by assigning their area names.

**Grid-auto-rows**: defines the size for all rows without a specified height, ensuring that they are all equal sizes.

1. **Grid Template Definitions**:
   * Create a sample grid layout using **grid-template-columns** and **grid-template-rows** properties. Include at least three columns and two rows in your example.

A screen shot of a computer program

Description automatically generated

* Describe the purpose of these properties in defining the grid structure.

The purpose of these properties is to define the number of rows/columns there are in the grid. In this example above, I defined three columns and had six items, meaning that the browser knew that there should be 2 rows.

**Part 3: Applying Flexbox within Grid Layout**

1. **Flexbox Integration**:
   * Explain why combining Flexbox with Grid Layout can be beneficial for creating flexible and responsive designs.

Grid layout is good for the container for the whole webpage, however Flexbox is really powerful for styling the content inside of the grid cells.

* Provide an example of how you can use Flexbox properties, such as **display: flex**, within a grid item to control its content.

A screen shot of a computer

Description automatically generated

**Part 4: Implementing Grid Layout with ChatGPT**

1. **Project Setup**: Create a new HTML and CSS file for your responsive web page project in your chosen text editor, like VSCode.
2. **Design Requirements**: Define the design requirements for your web page, and discuss your ideas with ChatGPT. Ensure that your design includes at least two sections with content that gracefully adapts to different screen sizes.
3. **Grid Layout Implementation**: Use ChatGPT's guidance to implement Grid Layout in your web page. Create a grid container and position grid items to represent your sections and other elements.
4. **Flexbox Integration**: Incorporate Flexbox properties within your Grid Layout, guided by ChatGPT, to ensure flexible content arrangement within each section.
5. **Remember:** You are still new to building Responsive Web Sites so you are not expected to build a professional looking site but one that includes all of the requirements.

**Part 5: Testing and Presentation**

1. **Testing**: Independently test your web page on various screen sizes (e.g., desktop, tablet, and mobile) to ensure responsiveness. If you encounter challenges, consult ChatGPT for troubleshooting tips.

**Submission:**

Submit your completed HTML and CSS files along with a brief written explanation of your design choices, the role of Grid Layout and Flexbox in your project, and how ChatGPT assisted you in learning and problem-solving. This should be submitted to GitHub. You can submit all of your answers to the questions as a document in document and upload it to GitHub with your HTML and CSS file.

On the website in the file index.html, my design choices were straightforward. I made the navbar using only flex, since I didn’t see a point of using grid for the navbar. I then made 2 columns, one each having an image and some text. The reason I didn’t use a grid of 2 columns and 2 rows is because then I wouldn’t have as much freedom with the size of the images. When the screen size is smaller, I then turned the navbar into a square and made the grid have 2 rows and one column, making it have the same formatting. ChatGPT helped my learning by giving script examples of how the grid layout works on HTML and CSS. I then created the whole final website by myself with no help from GPT whatsoever.

**Due Date: September 19th (End of Class)**