CS 104: Web Technology - II Assignment #1

Anup Adhikari
anup.adhikari@gandakiuniversity.edu.np

Gandaki University December 14, 2022

1 Introduction

We explored some properties of FLEX in previous example which could be used for single dimensional css layout system.

Lets explore the GRID Model in detail where we position different elements in two-dimensional space.

1.1 Grid Model

The CSS module defines a two-dimensional grid-based layout system, optimized for user interface design. In the grid layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout grid.

Grid Layout is a new layout model for CSS that has powerful abilities to control the sizing and positioning of boxes and their contents. Unlike Flexible Box Layout, which is single-axis-oriented, Grid Layout is optimized for 2-dimensional layouts: those in which alignment of content is desired in both dimensions.

1.1.1 Background

As websites evolved from simple documents into complex, interactive applications, techniques for document layout, e.g. floats, were not necessarily well suited for application layout. By using a combination of tables, JavaScript, or careful measurements on floated elements, authors discovered workarounds to achieve desired layouts. Layouts that adapted to the available space were often brittle and resulted in counter-intuitive behavior as space became constrained. As an alternative, authors of many web applications opted for a fixed layout that cannot take advantage of changes in the available rendering space on a screen.

The capabilities of grid layout address these problems. It provides a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Authors can then precisely position and size the building block elements of their application into the grid areas defined by the intersections of these columns and rows.

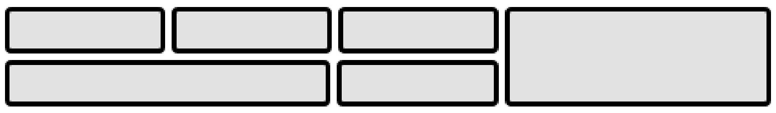

Figure 1: Representative Grid Layout Example

1.1.2 Adapting Layouts to available Space

Grid layout can be used to intelligently resize elements within a webpage. The adjacent figures represent a game with five major components in the layout: the game title, stats area, game board, score area, and control area. The author's intent is to divide the space for the game as shown in Figure 1.1.2 such that:

- The stats area always appears immediately under the game title.
- The game board appears to the right of the stats and title.

- The top of the game title and the game board should always align.
- The bottom of the game board and bottom of the stats area align when the game has reached its minimum height. In all other cases the game board will stretch to take advantage of all the space available to it.
- The controls are centered under the game board.
- The top of the score area is aligned to the top of the controls area.
- The score area is beneath the stats area.
- The score area is aligned to the controls beneath the stats area.

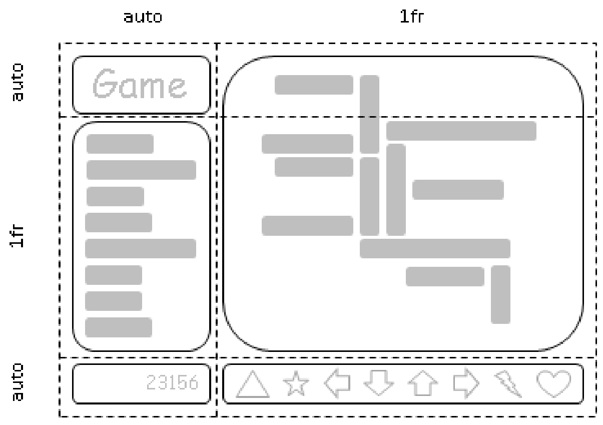

Figure 2: Five Grid items arranged according to content size and available space

```
<html>
    <head>
        <title >Grid Layout</title >
    </head>
    <body>

    </body>
</html>

<style >
#grid {
  /**
   * Two columns:
   *   1. the first sized to content,
   *   2. the second receives the remaining space
   *      (but is never smaller than the minimum size of the board
   *      or the game controls, which occupy this column [Figure 4])
   *
   * Three rows:
   *   3. the first sized to content,
   *   4. the middle row receives the remaining space
   *      (but is never smaller than the minimum height
   *       of the board or stats areas)
   *   5. the last sized to content.
   */
  display: grid;
  grid-template-columns:
    /* 1 */ auto
    /* 2 */ 1fr;
  grid-template-rows:
    /* 3 */ auto
    /* 4 */ 1fr
    /* 5 */ auto;
}

/*  The Fr Unit : Fr is a fractional unit.
    The Fr unit is an input that automatically calculates layout divisions
    when adjusting for gaps inside the grid. */
/* Specify the position of each grid item using coordinates on
 * the 'grid-row' and 'grid-column' properties of each grid item.
 */
#title    { grid-column: 1; grid-row: 1; }
#score    { grid-column: 1; grid-row: 3; }
#stats    { grid-column: 1; grid-row: 2; align-self: start; }
#board    { grid-column: 2; grid-row: 1 / span 2; }
#controls { grid-column: 2; grid-row: 3; justify-self: center; }
</style>

<div id="grid">
  <div id="title">Game Title</div>
  <div id="score">Score</div>
  <div id="stats">Stats</div>
  <div id="board">Board</div>
  <div id="controls">Controls</div>
</div>
```

The following example uses grid layout's ability to name the space which will be occupied by a grid item. This allows the author to avoid rewriting rules for grid items as the grid's definition changes.

```
<style>
#grid {
    display: grid;

    /* The rows, columns and areas of the grid are defined visually
     * using the grid-template-areas property.  Each string is a row,
     * and each word an area.  The number of words in a string
     * determines the number of columns. Note the number of words
     * in each string must be identical. */
    grid-template-areas: "title stats"
                         "score stats"
                         "board board"
                         "ctrls ctrls";

    /* The way to size columns and rows can be assigned with the
     * grid-template-columns and grid-template-rows properties. */
    grid-template-columns: auto 1fr;
    grid-template-rows: auto auto 1fr auto;
  }

/* The grid-area property places a grid item into a named
 * area of the grid. */
#title    { grid-area: title }
#score    { grid-area: score }
#stats    { grid-area: stats }
#board    { grid-area: board }
#controls { grid-area: ctrls }
</style>
<body>
  <div id="grid">
    <div id="title">Game Title</div>
    <div id="score">Score</div>
    <div id="stats">Stats</div>
    <div id="board">Board</div>
    <div id="controls">Controls</div>
  </div>
</body>
```

Info: Do you want to learn more on GRID?
Visit: https://cssgridgarden.com/

Question 1

Write the Grid layout to render the html file with desired format as in Figure 3

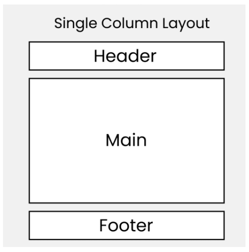

Figure 3: Col-Layout Grid

Question 2

Write the Grid layout to render the html file with desired format as in Figure 4

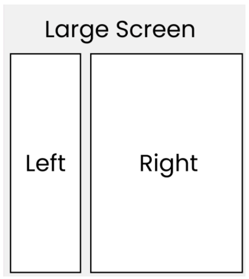

Figure 4: Col-Layout Grid

Question 3

Write the Grid layout to render the html file with desired format as in Figure 5

Question 4

Write the Grid layout to render the html file with desired format as in Figure 6

Question 5

Write the Grid layout to render the html file with desired format as in Figure 7

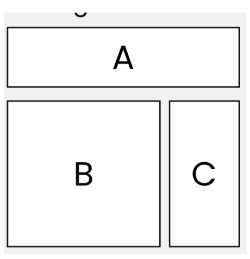

Figure 5: Col-Layout Grid

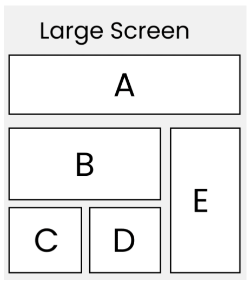

Figure 6: Col-Layout Grid

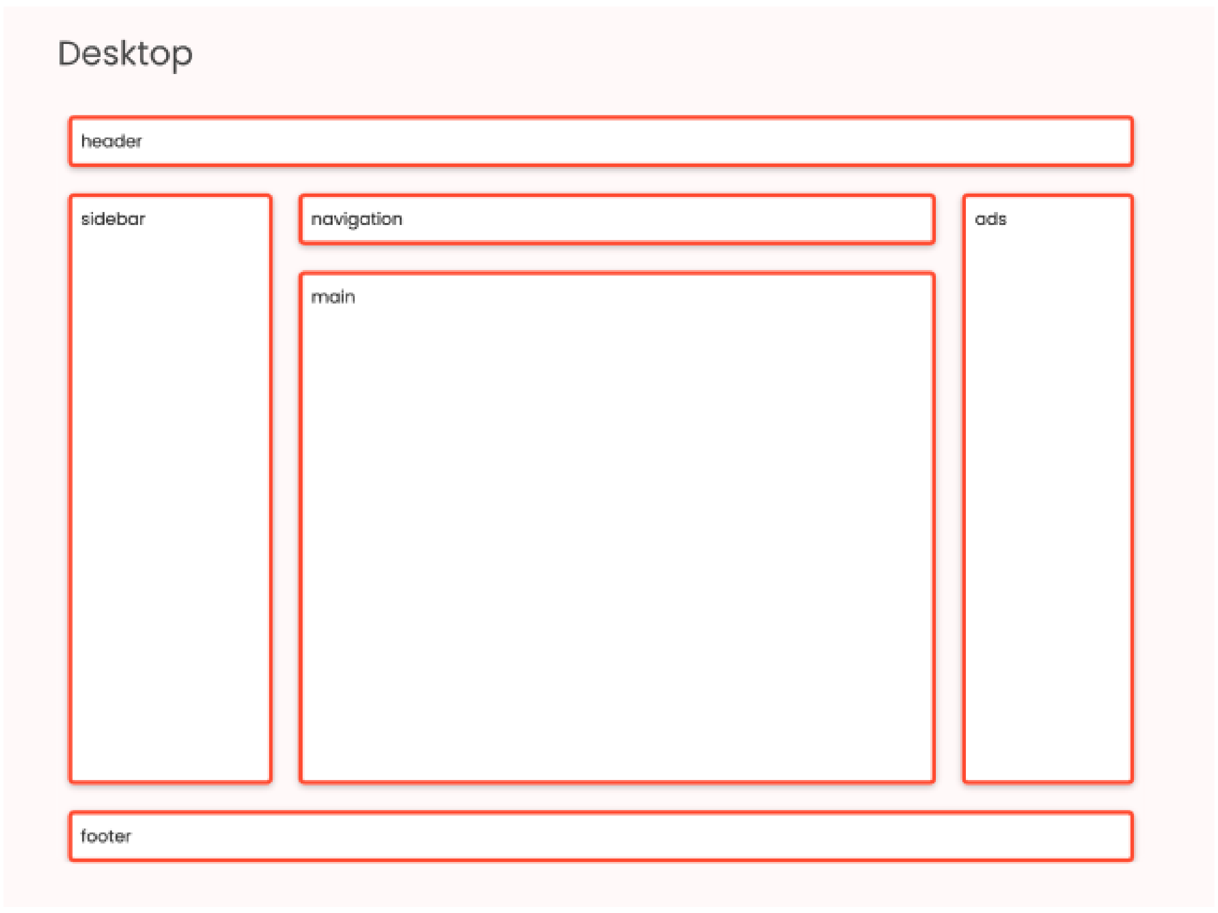

Figure 7: Suming Up Questions