International Journal on Recent and Innovation Trends in Computing and Communication
Volume: 3 Issue: 1

ISSN: 2321-8169
284 - 287

Grid Layout – A Fluid Web Framer

Gaurav Vaswani
Student,
Third Year Degree,
Computer Engineering,
VESIT, Chembur, Mumbai.
gauravvaswani@gmail.com

Priyanka Punjabi
Student,
Third Year Degree,
Computer Engineering,
VESIT, Chembur, Mumbai.
priyankapunjabi94@gmail.com

Ajay Chotrani
Student,
Third Year Degree,
Computer Engineering,
VESIT, Chembur, Mumbai
chotrani.ajay@gmail.com

Abstract :- Building the website with the table based design is a very common technique to structure the data. CSS does not bring forward any structure layout for rows and columns. The new fluid layout allows the even distribution of rows and columns in the form of grid structure. The new CSS grids resemble the way of table-based designs quite closely. The new grid fluid layout allows the designer to break the conventional use of table layout to properly distribute the contents on the web. This paper presents the basic layout structure of grid layout for web designing.

Keyword: *Grid layout, frames, HTML, CSS,grid tracks, grid lines.*

*****

I. INTRODUCTION

Developing websites from simple documents to the visual layouts to well suited application structure is the necessity of the web environment. With the combinations of tables, JavaScript and the various floating elements it was difficult to achieve the desired output.The layout structure developed by W3 is the layout structure which helps to design and that adapted to the available space were often brittle and resulted in counter-intuitive behavior as space became constrained.

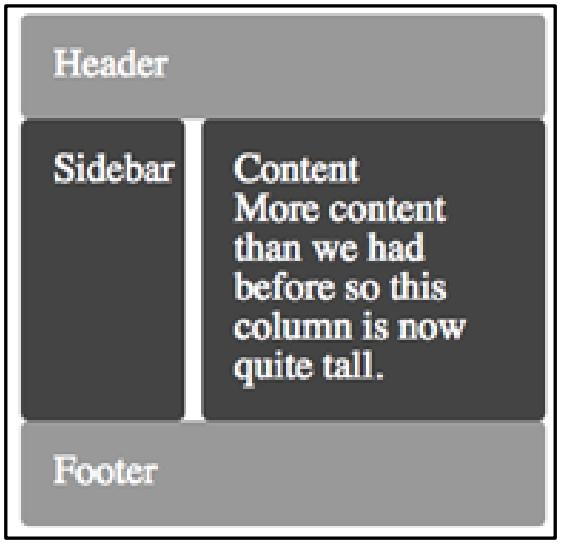

Figure 1: Application layout example requiring horizontal and vertical alignment.

The capabilities of grid layout address problems. It provides a mechanism for designers to develop articulate and to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Web developers can then precisely position and size the building block elements of their application by into grid areas defined by these columns and rows and develop the required application. Figure 1 illustrates a basic layout which can be achieved with grid layout.

ADAPTING LAYOUTS TO AVAILABLE SPACE

Grid layout can be used to intelligently reflow elements within a webpage in the form of rows and columns. Figure 2 header, sidebar, content, slide bar and the footer in the five layout structure.

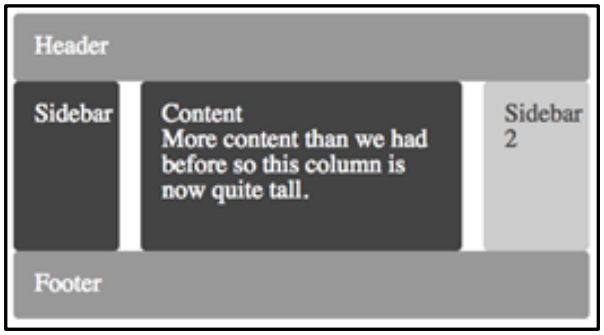

Figure 2: Five grid items arranged according to content size and available space.

The web developer can design the web area as

- The stats area Header can always appears with the title of the project or the application.
- The sidebar can show the stats and the title of the project.
- The side bar, content, and the slide bar are always aligned.
- The bottom of the game board and the stats area shows the footer for the web alignment and in line with the header.

As an alternative to using script to control the absolute position, width, and height of all

There are multiple ways to specify the structure of the grid and to position and size grid items, each optimized for different web development application scenarios. This example illustrates the use to define the position and space for each grid item using thegrid-template-rows and grid-template-columns properties on the grid container, and the grid-row and grid-column properties on each grid item. The example taken is from the W3.org which shows the absolute position of the growth in the row and the column.

ISSN: 2321-8169
280 - 283

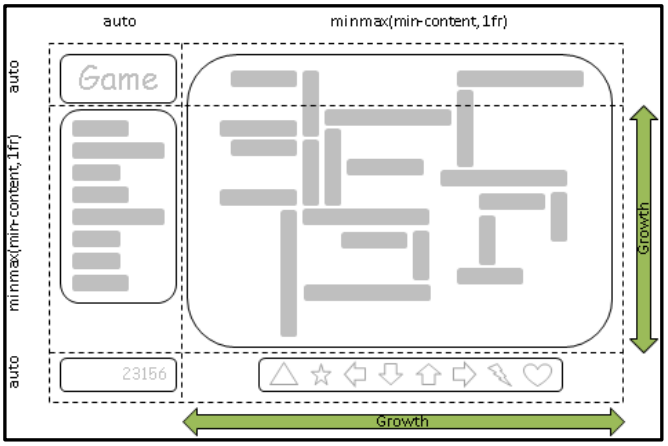

Figure 3: Absolute position

II. SOURCE INDEPENDENCE

The example taken from the above reference of Figure 3 is implemented for the game to adopt to the various spaces space available on traditional computer monitors, handheld devices, or tablet computers. The placement optimize the components when viewed either in landscape or portrait orientation (Figures 4 and 5). By combining grid layout with media queries, to use the same semantic markup, but rearrange the layout of elements independent of their source order, to achieve the desired layout in both orientations.

The following example emphasizes grid layout's ability to name the space which will be occupied by a grid item. This gives the web developer an advantage to avoid rewriting rules for grid items as the grids

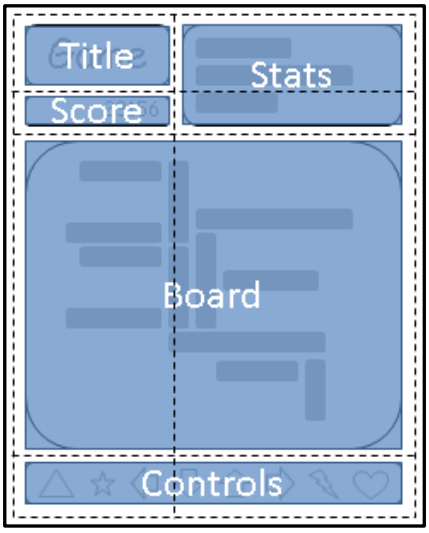

Figure 4: An arrangement suitable for portrait orientation.

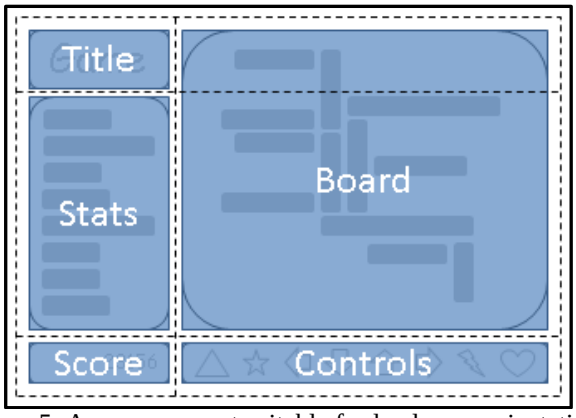

Figure 5: An arrangement suitable for landscape orientation.

GRID LAYERING OF ELEMENTS

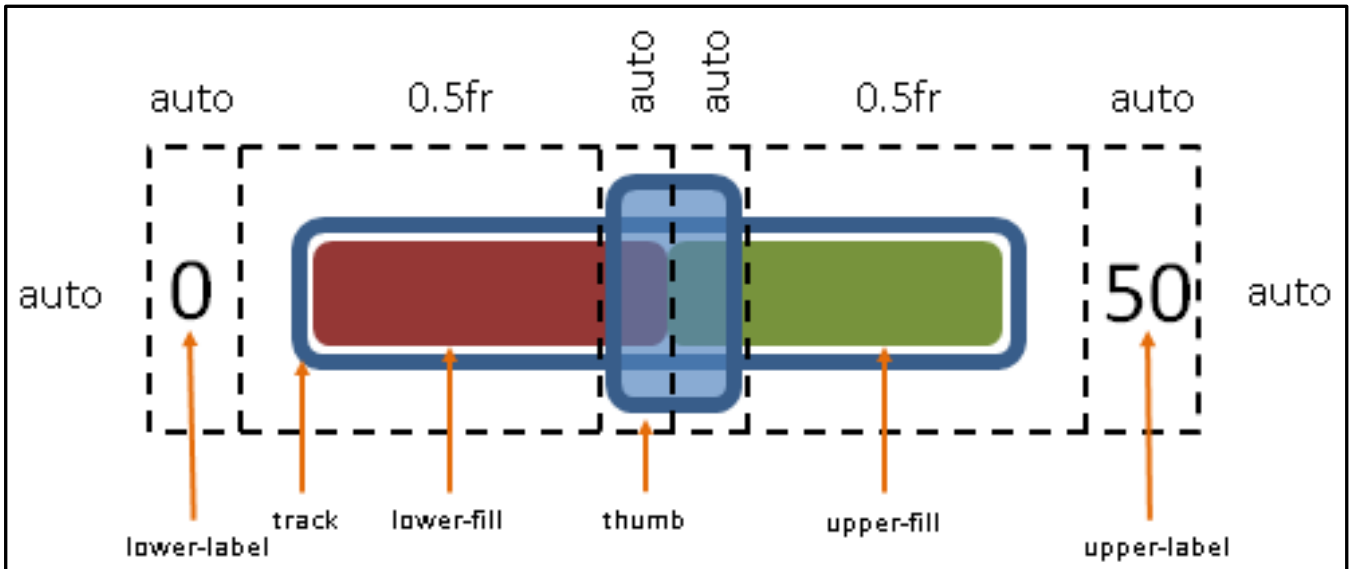

Figure 6: Grid layering of elements

 ISSN: 2321-8169
280 - 283

A control composed of layered HTML elements.

In the example shown in Figure 7, the custom layout slider control for the three in block axis and four in line axis structure. The control has six parts. The lower and upper labels align to the left and right edges of the control. The track of the slider spans the area between the labels. The lower and upper fill parts touch beneath the thumb, and the thumb is a fixed width and height that can be moved along the track by updating the two flex-sized columns.

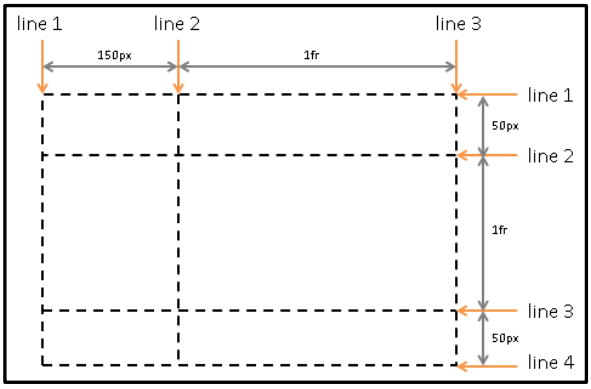

Figure 7: Grid lines: Three in the block axis and four in the inline axis.

The absolute positioning control the top and left coordinates, along with the width and height of each HTML element that comprises the control. By taking the advantage of the grid layout, the web designer can instead limit script usage to handling mouse events on the thumb, which snaps to various positions along the track as the grid-template-columns property of the grid container is updated.

GRID LAYOUT CONCEPTS AND TERMINOLOGY

The structural layout of the grid structure comprises of section, col, and group as explained

.section

Splits up the page horizontally. Need a few of these to break up the content, and you can use them in your main wrapper, or within other divisions.

.col

Divides the section into columns. Each column has a left margin of 1.6% (around 20 pixels on a normal monitor), except the first one. Using .col:first-child { margin-left: 0; } means you don't need to use class="last" anywhere.

.group

This solves floating problems, by forcing the section to self-clear its children (aka the clearfix hack).

Grid Tracks and Cells

Grid track is a generalized structure for a grid column or grid row, it is also considered as the space between two adjacent grid lines. Each grid track is assigned a sizing function, which controls how wide or tall the column or row may grow, and thus how far apart it's bounding grid lines are.

A grid cell the full grid which specifies and it is the space between two adjacent row and two adjacent column grid lines. It is the smallest unit of the grid that can be referenced when positioning grid items.

In the following example there are two columns and three rows. The first column is fixed at 200px. The second column uses flexible sizing, which is a function of the unassigned space in the Grid, and thus will vary as the width of the grid container changes. If the used width of the grid container is 250px, then the second column 50px wide. If the used width of the grid container is 100px, then the second column is 0px and any content positioned in the column will overflow the grid container.

```
<style type="text/css">
   #grid {
display: grid;
grid-template-columns: 200px 1fr; /* two columns */
grid-template-rows: 50px 1fr 50px /* three rows  */
   }
</style>
```

Grid Lines

Grid lines as the name specifies are the horizontal and vertical dividing lines of the grid. A grid lines always exists on either side of a column or row. They can be referred to by numerical index, or by an author-specified name. A grid item references the grid lines to determine its position within the grid using the grid-placement properties.

The following two examples create three column grid lines and four row grid lines. The first example demonstrates how the web developer would position a grid item using grid linenumbers. The second example uses explicitly named grid lines.

```
<style type="text/css">
   #grid {
display: grid;
grid-template-columns: 200px 1fr;
grid-template-rows: 50px 1fr 50px
   }
```

 ISSN: 2321-8169
280 - 283

```
#item1 { grid-column: 2;
grid-row-start: 1; grid-row-end: 4; }
</style>
<style type="text/css">
    /* equivalent layout to the prior example, but using named lines */
    #grid {
display: grid;
grid-template-columns: 200px (item1-start) 1fr (item1-end);
grid-template-rows: (item1-start) 50px 1fr 50px (item1-end);
    }

    #item1 {
grid-column: item1-start / item1-end;
grid-row: item1-start / item1-end
    }
</style>
```

Grid Areas

A grid area is considered as the logical space used to lay out one or more grid items. It is bounded and developed between the by four grid layout structural lines, one on each side of the grid area, and participates in the sizing of the grid tracks it intersects. A grid area can be named explicitly using the grid-template-areas property of the grid container, or referenced implicitly by its bounding grid lines. A grid itemis assigned to a grid area using the grid-placement properties.

```
<style type="text/css">
  /* using the template syntax */
  #grid {
  display: grid;
  grid-template-areas: “ a"
                       "b a"
                       ". a";
  grid-template-columns: 150px 1fr;
  grid-template-rows: 50px 1fr 50px
  }

  #item1 {grid-area: a}
  #item2 {grid-area: b}
  #item3 {grid-area: b}

  /* align items 2 and 3 at different points in the Grid Area "b". */
  /* by default, Grid Items are stretched to fit their Grid Area */
  /* and these items would layer one over the other. */
  #item2 { align-self: head}
  #item3 { justify-self: end; align-self: foot }
</style>
```

III. AUTOMATIC GRID ITEM PLACEMENT ALGORITHM

The following auto-placement algorithm places grid items with automatic grid positions into a definite grid position.

For the purpose of this algorithm, an occupied grid cell is any grid cell which is contained within any named grid area or the grid area of any previously-positioned grid item.

1. Position anything that’s not auto-positioned.Before auto-positioning anything, position every grid item with a definite grid position in both axes.
2. Process the items locked to a given row. For each grid item with a definite row position (that is, the grid-row-start and grid-row-end properties define a definite grid position), in order-modified document order, position its inline-start edge to the earliest (smallest positive index) line index that ensures this item’s grid area will not overlap any occupied grid cells.
3. Determine the number of columns in the implicit grid.

Set the number of columns in the implicit grid to the larger of:

- The number of columns in the explicit grid.
- Among all the items with a definite column position (explicitly positioned items, items positioned in the previous step, and items not yet positioned but with a definite column) the largest positive line index that an item’s inline-end edge is positioned to, minus 1.
- Among all items with a definite column span, the largest such span.

For example, in the following style fragment:

```
 #grid {
display: grid;
grid-template-columns: repeat(5, 100px);
grid-auto-flow: row;
  }
  #grid-item {
    grid-column: 4 / span 3;
  }
```

The number of columns needed is 6. The #grid-item element’s inline-start edge is positioned at index 4, so its span ensures that its inline-end edge will be positioned at index 7. This requires 7 - 1 = 6 columns to hold, which is larger than the explicit grid’s 5 columns.

4. Position the remaining grid items.

The auto-placement cursor defines the current "insertion point" in the grid, specified as a pair of row and column grid lines. Initially the auto-placement cursor at is specified with a row and column position both equal to 1.

 ISSN: 2321-8169
280 - 283

For each grid item that hasn't been positioned by the previous steps, in order-modified document order:

IV. CONCLUSION

The fluid grid layout provides an easy scope for the developers to organize their data. The visual structure and the CSS 3 coding is precisely implemented. This feature enhances the web development application.Layout in CSS is only getting more powerful, and flexbox is one of the first steps out of the quagmire we've found ourselves in over the years, first with table-based layouts, then float-based layouts.

REFERENCES

[1] http://www.w3.org/TR/css-grid-1/
[2] http://www.webdesignbizz.com/
[3] http://www.creativebloq.com/css3/create-fluid-layouts-html5-and-css3-3142768.
[4] http://www.noupe.com/design/html5-brings-tables-back-contemporary-grid-layouts-with-css-grids-83471.html.
[5] http://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html.
[6] http://webdesign.tutsplus.com/articles/all-about-grid-systems--webdesign-14471.
[7] http://webdesign.tutsplus.com/articles/improving-layout-with-vertical-rhythm--webdesign-14070.
[8] http://css-tricks.com/snippets/css/ complete-guide-grid/