CS 104: Web Technology - II Assignment #1

Anup Adhikari
anup.adhikari@gandakiuniversity.edu.np

Gandaki University November 30, 2022

1 Title: Advanced CSS Using Flex and Grid Properties in Front-End Web Design

1.1 Objectives

The objective of the lab is:

- Demonstrate the techniques of exploring different designs provided by flex and grid.
- Identify the proper selection of display properties for modern CSS.

1.2 Theory

1.2.1 Flex Model

In the flex layout model, the children of a flex container can be laid out in any direction, and can "flex" their sizes, either growing to fill unused space or shrinking to avoid overflowing the parent. Both horizontal and vertical alignment of the children can be easily manipulated. Nesting of these boxes (horizontal inside vertical, or vertical inside horizontal) can be used to build layouts in two dimensions.

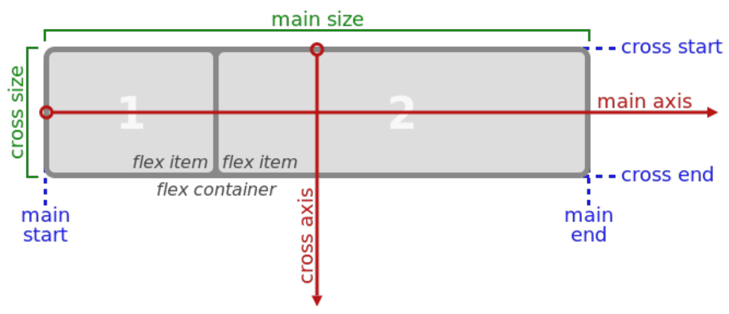


Figure 1: Terms

What can be done using flex property?

- can be laid out in any flow direction (leftwards, rightwards, downwards, or even upwards!)
- can have their display order reversed or rearranged at the style layer (i.e., visual order can be independent of source and speech order)
- can be laid out linearly along a single (main) axis or wrapped into multiple lines along a secondary (cross) axis
- can "flex" their sizes to respond to the available space
- can be aligned with respect to their container or each other on the secondary (cross)
- can be dynamically collapsed or uncollapsed along the main axis while preserving the container's cross size

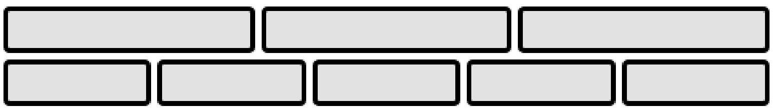


Figure 2: Representative Flex Layout Example

1.2.1.1 Flex Properties

Following are the key terms to understand in flex:

1. Flex Items

   Info: The margins of adjacent flex items do not collapse. They paint items exactly like inline-blocks. Order modified-document 'order' is used in place of raw document order. Specifying 'visibility:collapse' on flex item causes it to become a collapsed flex item.

2. Ordering and Orientation

   - flex-direction
   - flex-wrap
   - flex-flow
   - order

3. Flex lines:

   Flex items in a flex container are laid out and aligned within flex lines, hypothetical containers used for grouping and alignment by the layout algorithm. A flex container can be either single-line or multi-line, depending on the flex-wrap.

4. Flexibility

   - flex-grow
   - flex-shrink
   - flex-basis

5. Alignment

Example 1 - CSS Flex

```
#deals {
display: flex;         /* Flex layout so items have equal height  */
flex-flow: row wrap;   /* Allow items to wrap into multiple lines */
}
.sale-item {
display: flex;         /* Lay out each item using flex layout */
flex-flow: column;     /* Lay out item's contents vertically  */
}
.sale-item > img {
order: -1;             /* Shift image before other content (in visual order) */
align-self: center;    /* Center the image cross-wise (horizontally)         */
}
.sale-item > button {
margin-top: auto;      /* Auto top margin pushes button to bottom */
}
```

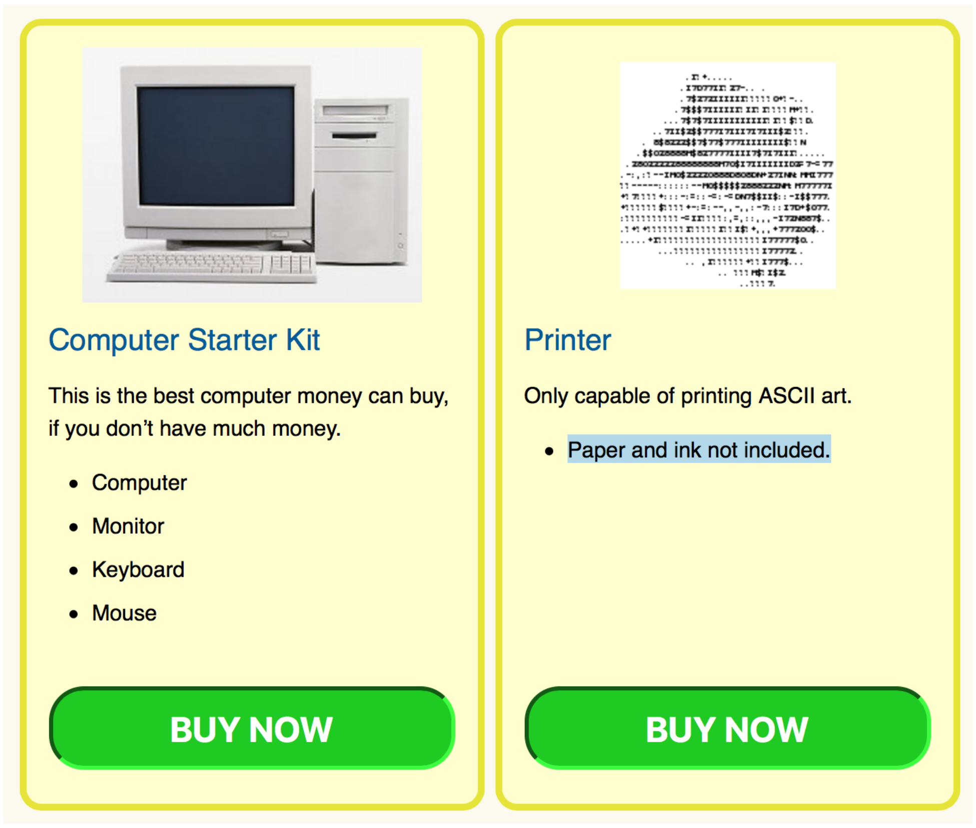
Example 1 - HTML Flex

```html
<section id="deals">
    <section class="sale-item">
        <h1>Computer Starter Kit</h1>
        <p>This is the best computer money can buy,
            if you don't have much money.</p>
        <ul>
        <li>Computer</li>
        <li>Monitor</li>
        <li>Keyboard</li>
        <li>Mouse</li>
        </ul>
        <img src="images/computer.jpg"
            alt="You get: a white computer with matching peripherals.">
        <button>BUY NOW</button>
    </section>
    <section class="sale-item">
        <h1>Printer</h1>
        <p>Only capable of printing ASCII art.</p>
        <ul>
        <li>Paper and ink not included.
        </ul>
        <img src="images/printer.jpg"
            alt="You get: a printer with matching details.">
        <button>BUY NOW</button>
    </section>
</section>
```

1.2.2 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.



Computer Starter Kit

This is the best computer money can buy, if you don't have much money.

- Computer
- Monitor
- Keyboard
- Mouse

Printer

Only capable of printing ASCII art.

- Paper and ink not included.

Figure 3: Output of Example 1

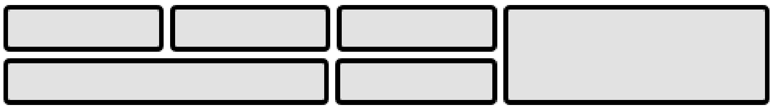


Figure 4: Representative Grid Layout Example

Example 2 - CSS Grid

```
/**
* Define the space for each grid item by declaring the grid
* on the grid container.
*/
#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;
}
```

Example 2 - HTML Grid

```
<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>
```

Question 1

Write the CSS to display the webpage to have collapsed properties as that of Figure 5.

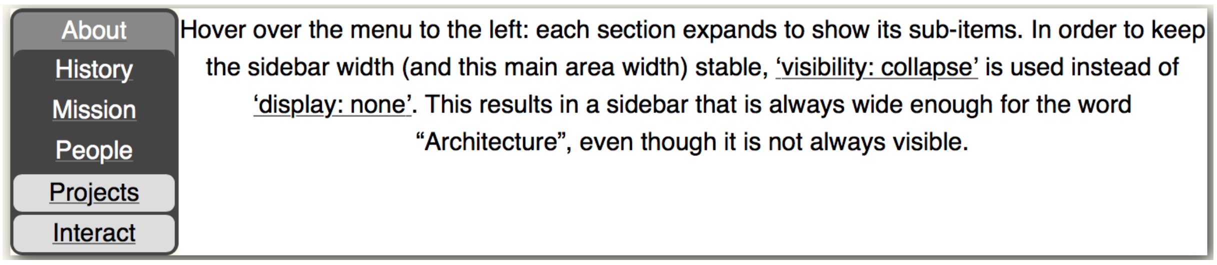
File

```
<div>
    <article id="main">
        Interesting Stuff to Read
    </article>
    <nav>
        <ul>
        <li id="nav-about"><a href="#nav-about">About</a>
            ...
        <li id="nav-projects"><a href="#nav-projects">Projects</a>
            <ul>
            <li><a href="..." >Art</a>
            <li><a href="..." >Architecture</a>
            <li><a href="..." >Music</a>
            </ul>
        <li id="nav-interact"><a href="#nav-interact">Interact</a>
            ...
        </ul>
    </nav>
</div>
<footer> </footer>
```

Info: dynamically collapse submenus when not targetted
nav > ul > li:not(:target):not(:hover) > ul {
visibility: collapse;}

Question 2

Write CSS and html to display the web content as shown in figure 6.



Hover over the menu to the left: each section expands to show its sub-items. In order to keep the sidebar width (and this main area width) stable, 'visibility: collapse' is used instead of 'display: none'. This results in a sidebar that is always wide enough for the word "Architecture", even though it is not always visible.

Figure 5: Collapsible Sidebar

Question 3

Create a chess grid. (12x12).

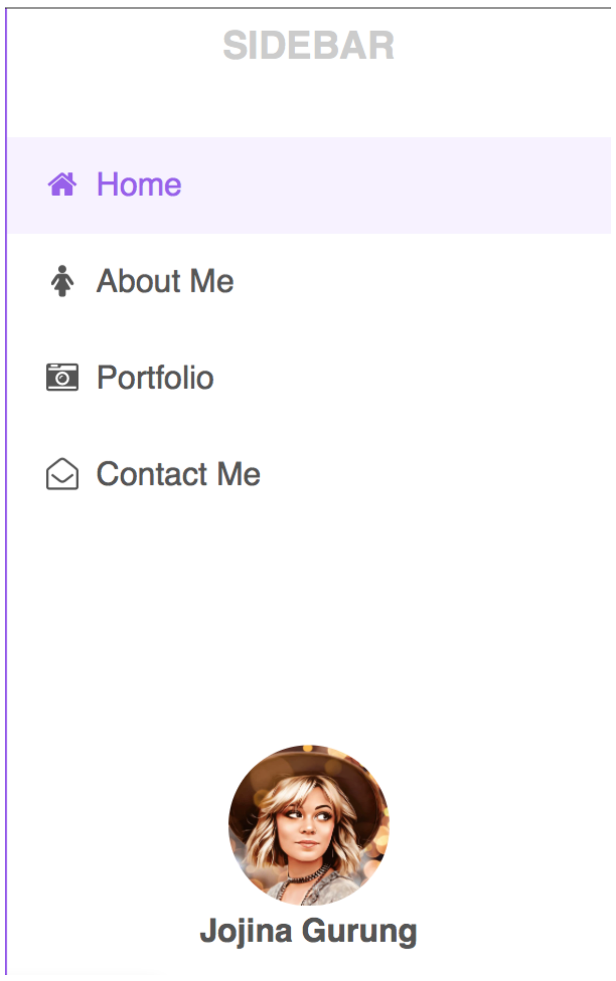


Figure 6: Exercise with Flex