**CSS Columns** is a straightforward way to style content so it is gathered together into 2 or more columns instead of just flowing horizontally to the edge of the parent element. This mimics a traditional printed layout and is especially intuitive for displaying text, since it avoids the long and narrow paragraph look when displaying text on wide screens.

Columns is limited in what it can accomplish and leaves many of the decisions of how to fit the parameters up to the browser. It is easy to group text or elements into several columns across the horizontal page axis, but doing more than a basic newspaper layout is difficult. Because you define max number of columns and minimum column width, the final appearance will be determined by the browser trying to fit within that specification. Additionally, some of the advanced parameters, such as those that control where column breaks occur, don’t work in Firefox.

**Flexbox** is a simple way to create page layouts that adjust to a variety of screen sizes and layouts and can handle diverse child elements. A web developer can individually define how to handle sizing, spacing, alignment, and wrapping of elements, and they can control positioning on both the horizontal and vertical axes to easily handle various screen orientations.

Flexbox is a great way to easily place multiple items of arbitrary size along one axis of a parent element. For one, no item can span multiple tracks in the secondary axis, so combining items of different “heights” can be tricky (without using wrappers and nested flexboxes). Second, while there are many spacing and justification options, they lack precision and usually are used to try to evenly space items. This can be useful, but makes it hard to carefully position items in specific regions.

**CSS Grid** is a powerful and flexible way to define layout along both page axes. The designer defines a template for both rows and/or columns, then can position children arbitrarily within that template, either as individual cells or as multi-cell regions. Templates can use sizes that are absolute, proportional, or a mixture, allowing many possible options.

Grids take a bit more effort to define, but allow for much more precise control of placement than other responsive design strategies. For simple cases, the designer can still trust the “auto” value to handle the details, and can mix that with specific values for other rows or columns seamlessly.

**Combining Grid and Flexbox** is a particularly useful option, because it allows you to get “the best of both worlds” in many cases. The strength of Grid is the ability to carefully define absolutely positioned elements and to mix that with fluid regions, while flexbox focuses on letting the browser figure out the best way to fit an arbitrary collection of items into a defined space. Thus, you can use Grid to frame out the page’s overall structure, allocating space for things like navbars and other major elements, while using flexbox for an arbitrary number of children within these regions.

**The “Best” Layout:**

While there’s obviously not a single “right” way to handle layout for all pages, CSS Grid offers a compelling model that is powerful and flexible, and web developers would do well to seriously consider it for modern responsive projects. While it requires a bit more effort to set up the grid template than methods like flexbox, the ability to seamlessly weave hand-tuned elements alongside proportional content areas is useful in a wide range of scenarios. Additionally, using Grid does not preclude leveraging the strengths of other strategies, particularly Flexbox, within specific regions of the page where appropriate.