CSS Grid Layout: A Revolution in Web Layout Design

Timo Köster
Fachhochschule Südwestfalen

***Abstract*—This paper gives the reader a quick summary of older approaches to designing layouts for the web. It argues that the introduction of *CSS Grid Layout* revolutionized the way web designers approach the layout of a website. Its advantages and ease of use have ushered in a new era of beautifully designed layouts for websites. The reader will find visual examples, a code listing for every older approach and an introduction to the grid layout in print media, as well as the core concepts of *CSS Grid Layout*.**

I. Introduction

With the launch of the iPhone in 2007 [1] mobile devices that are able to access the web were made available to a broad base of customers for the first time. This trend gained even more momentum with the introduction of the iPad in 2010 [2] and of Android phones starting in 2008 [3]. In 2015 Google stated that, for the first time, the amount of searches using mobile devices was higher than the amount of searches via desktop devices [4]. This trend had continued. In 2022 the share of website traffic coming from mobile devices was at 58.33% excluding tablets [5]. As mobile devices usually have different requirements for web pages than desktop computers the paradigm of mobile first was introduced by Google in 2016 [6]. In 2020, this became a core measurement for ranking scores by Google for every website [7]. Besides requirements for speed and performance one of the major paradigm shifts was layout. Layouts now need to be responsive. That means websites have to be accessible and appealing on screens of different sizes. Older approaches to layout, such as tables, floats and *Flexbox*[1] are not easy to handle. They need a lot of specialized code for various device sizes[2]. The widespread adaption of *CSS Grid Layout* starting in 2017 [8] revolutionized designing responsive layouts. But what is so revolutionary about *CSS Grid Layout*?

In this article I will show what is so innovative about *CSS Grid Layout*. For this purpose, some examples of different older approaches to web layout design will be given. Their respective disadvantages in a modern web environment will be shown. I will take a look at the core functionalities of the *CSS Grid Layout* and its benefits in web layout design. The conclusion will expound on reasons why some aspects of classic web layout technologies are still relevant in certain instances. I will show why *CSS Grid Layout* is the first choice in modern web layouts for most use cases. All figures were created using *Inkscape 1.3* or using the screenshot function of *Mozilla Firefox 120.0.1*. Most listings are excerpts. The full code is available in a repository[3].

II. Grid Layout in print

Since the invention of printing with movable types in the 1450s the ideas and theories of typography and layout have undergone many changes. Our "Modern typography is based primarily on the theories and principles of design [that] evolved in the 20's and 30's [...]" of the 20th century [9, p. 7]. These principles, however contained very strict and rigid rules about conception and composition. They did not include the modern concept of a grid [9, p. 7]. The grid as a guideline for layouts was developed between 1945 and 1960, by the so called *Schweizer Schule*. The ambition was to invent a system that could be used universally for all layout purposes [10, p. 109]. In succeeding they paved the way for an unparalleled rise of a concept in typography: "Grid use began to dominate European and American design during and after the 1960s." [11, p. 17] But what is a grid exactly? And what is it used for? Josef Müller-Brockmann, a central figure in the development of the grid, envisioned the grid as a modular tool to achieve coherency between all pages of a book or a series of posters. Coherency is supposed to give the publication an identity and harmony [10, p. 109]. This means that the viewer or reader can identify a poster as part of a series at a first glance. But how does this work? To cite Müller-Brockmann himself: "The grid divides a two-dimensional plane into smaller fields [...]" [9, p. 11]. They "[...] may be the same or different size. The fields correspond in depth to a specific number of lines of text and the width of the fields is identical with the width of the columns." [9, p. 11] Hence, they "[...] are separated by an intermediate space so that on the one hand pictures do not touch each other and legibility is thus preserved and on the other that captions can be placed below the illustrations." [9, p. 11] So a grid helps the designer to construct a layout that has well weighted proportions between different elements of a page. There is a general harmony on the pages in itself and with

Dieser Beitrag entstand im Rahmen des Bachelor-Konferenzseminars *info@swf 2024*, das im Wintersemester 2023/24 vom Fachbereich Informatik und Naturwissenschaften der Fachhochschule Südwestfalen durchgeführt wurde.

[1]Flexbox is a CSS Module for design layouts. See Section IV Subsection D of this paper

[2]For detailed descriptions of the mentioned techniques see section IV

[3]https://anonymous.4open.science/r/CSS-Grid-Layout---A-Revolution-in-Web-Layout-Design/

the other pages of a book or print series [10, p. 109]. To put it simply: The grid "[...] introduces systematic order to a layout. [...] it ensures vital cohesion among visual elements, harmonizing them through the systems of spatial proportions and positioning logic it defines" [11, p. 11]. Although the grid has been around for well over 60 years in print it is still a very relevant tool today.

III. CSS Grid Layout

Grid layouts have been used in print for a long time. An implementation for CSS was originally envisioned by Bert Bos, one of the inventors of CSS. But it wasn't until 2011 that the need for a grid layout in web design was pressing enough. As Microsoft was planning to allow web technologies for app design on Windows 8 a team of developers for *Microsoft Intune* started working on an implementation for CSS [12]. A first draft for level 1[4] was released in 2011 and a first Candidate Recommendation Snapshot in 2016. The current Recommendation is from 2020 [14]. Level 2 introduced sub-grids. It was drafted in 2018 and the latest Recommendation was released in 2020 [15]. This article will reference the 2020 Recommendation Candidates.

A. Core Concepts of CSS Grid Layout

The *CSS Grid Layout* Module introduced some new core concepts to translate the very common Grid Layout technique to CSS. The following sections explain these core concepts.

1) Tracks and Gutters: CSS Grid Layout works with grid lines that form *tracks* and *gutters* within a grid container. The Grid Layout Module is used to achieve 2-dimensional alignments. This is in contrast to *Flexbox* (1-dimensional). The analogy of a table is helpful here [16]. A grid cell is at the intersection of a grid row and a grid column. Content can be placed here [16]. *Tracks* are divided into columns and rows and contain the grid items [17, p. 256]. This works similarly to a table. It can be said that a track is the area between two grid lines [18, p. 123]. Consisting of one or more adjacent grid cells a *grid area* can be named explicitly [16]. However, there does not necessarily have to be content in a "table cell". To avoid content being directly adjacent, so-called *gutters* can be defined. These are basically gaps between content, hence the properties `row-gap`, `column-gap` and `gap` [16]. The *gutters* are the equivalent to the intermediate space Müller-Brockmann mentions. A visual representation of *grid areas* and specifically *gutters* helps to clarify what he means and to put it into context with modern web design. In figure 1 the rectangle colored in a gradient is a *grid area* while the medium grey areas are *gutters*. A track row is colored dark grey and a track column light grey.

[4]W3C uses the word level to refer to releases of modules. For a more detailed description of the W3C terminology regarding CSS releases see [13]

2) Explicit and Implicit Grid: We can define a basic grid with *tracks* and *gutters* and place items. This is called the *explicit* grid as we *explicitly* define the *tracks* and the placement of the items [18, p. 123]. Sizes can be assigned to the rows and columns of an *explicit* grid. For this the attributes `grid-templatecolumns: <size>…;` and `grid-template-rows: <size>…;` are used, respectively. `grid-template-areas` is used to define a *grid area* [19, pp. 272-273]. The items can explicitly be placed in *tracks* or areas. But what happens with items that are added after the initial creation of the grid? The *Grid Layout Module* has a clever solution in store for these items: The *implicit* grid. This *implicit* grid is comprised of generated grid *tracks*. Items that are not placed in the *explicit* grid are placed in a track of the *implicit* grid. This happens automatically. Although the grid is *implicit* and may not be generated at the creation of the *explicit* grid, sizes of *tracks* can still be defined by the designer. For this purpose the `grid-auto-rows` and the `grid-auto-columns` properties need to be set. `grid-auto-flow` has to be included to activate the possible generation of an *implicit* grid. It is good practice to enable automatic placement. It places the items that are not placed *explicitly* and is activated if the *explicit* grid is full [16].

B. Example

Here is a simple example that shows *explicit*, *implicit* grid and responsiveness. Checkered Objects are placed in the *explicit* grid and striped objects implicitly placed in figure 2. In figure 3 the same objects are rearranged automatically to fit a smaller screen. The CSS in listing 1 shows what happens in the container class for this effect. `grid-template-columns` is set to `repeat()` and `auto-fill`. This means, that the column is repeated as many times as possible until the width of the page is full. If we decrease the size of the screen it rearranges the columns. This even works without *media queries*. The HTML basis can be seen in listing 2.

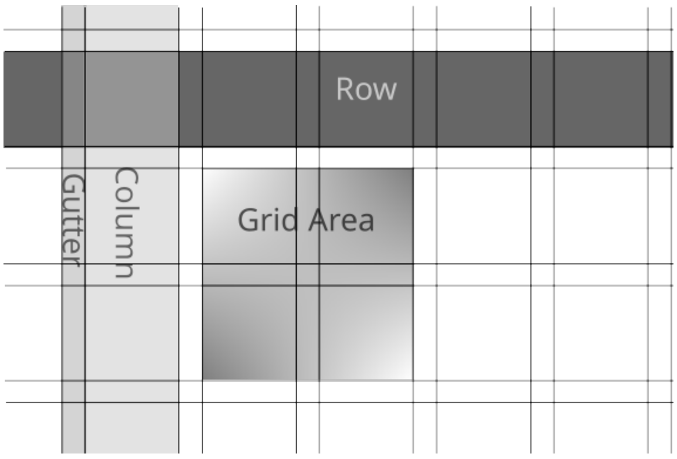


Figure 1. Schematics of *grid area* and *gutters*

IV. Older Approaches to Layouts of Websites

The most important techniques and approaches to layout design in web development that came before *CSS Grid Layout* will be shown here to demonstrate the advantages of *CSS Grid Layout*. This is by no means a complete overview of all techniques of the past, nor a deep dive into any of the techniques presented. For Examples of older methods of layout design see section V.

A. HTML and CSS

In the early stages of the World Wide Web the design options for the layout of websites were rather limited. In 1993, some browsers were available that mainly displayed text. But tables pictures and other features could be displayed by others, for example Arena [20, p. 67]. With the release of *Mosaic 0.9* for X on March 14th, 1993 that introduced inline positioning of pictures via the `<IMG>`-tag, layout became something to consider properly for the first time [21]. While Mosaic was a step back from earlier browsers in terms of customization abilities for style, it made the web widely popular. Many authors of websites complained about not being able to properly design a page. As HTML was never meant to be used to design websites other than on a structural level there was no convention on style sheets. And not every browser supported all HTML-tags like `<center>` or the like [12]. Due to this the layouts were often designed with tables. [17, p. 2] This practice is highly discouraged by the *W3C (World Wide Web Consortium)* today as tables are not meant to be used for layout [22]. The Introduction of CSS 1 in 1996 finally separated structure (HTML) and layout and Presentation (CSS) of a website as intended from the beginning [12]. CSS 1 introduced a wide range of techniques for styling a document. Not only choice of fonts or colors but also for designing a proper layout. It introduced a simple box model as shown in figure 4. These *block-level elements* like lists or paragraphs came with the possibility of not only positioning text within said block but also with the ability to use vertical and horizontal formatting of these blocks relative to one another [23]. The different boxes are created with `<div>`-tags. These div elements can be styled individually by using classes or IDs. The HTML-code can be seen in listing 3 and the associated CSS-code in listing 4. Another feature introduced in CSS 1, the *floats*, will be explained in the following section.

Listing 1. example1CSSGrid.html
```html
<!DOCTYPE html>
<html>
<head>
  <title>CSS Grid Layout Example</title>
  <link rel="stylesheet"
  href="example1CSSGrid.css">
</head>
<body>
  <div class="container">
    <div class="item"><img src="explicit.png">
    </div>
    <div class="item"><img src="explicit.png">
    </div>
    <div class="item"><img src="explicit.png">
    </div>
    <div class="item"><img src="implicit.png">
    </div>
    [...]
    <div class="item"><img src="implicit.png">
    </div>
  </div>
</body>
</html>
```

Listing 2. example1CSSGrid.css
```css
.container {
  display: grid;
  grid-template-columns: repeat(auto-fill, 20em)
      ;
  grid-template-rows: 1fr;
  grid-gap: 10px;
}
.item {
  background-color: #f2f2f2;
  padding: 20px;
  text-align: center;
}
```

B. Floated Layout

CSS 1 also introduced *floats* for positioning pictures and other elements outside of the flow of the structure of websites [23]. This means that a floating element floats within its parent element. Text and other elements float around it. Horizontal navigation bars were a common example of this. The technique is still used in some instances, for example images in blocks of text or pull quotes as shown in figure 6 [24, pp. 30-34]. However, they can "leak out" of the bottom of a relatively positioned element and impact the positioning of following relatively positioned elements [24, p. 30]. In other words the floating elements can overlap others and can greatly disrupt the flow of the site. An example can be found in the repository.

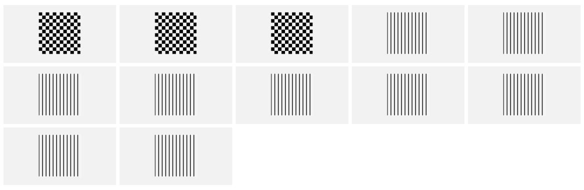


Figure 2. Items placed within *explicit* and *implicit* grid

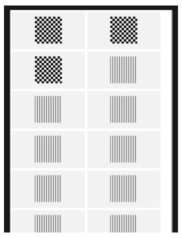
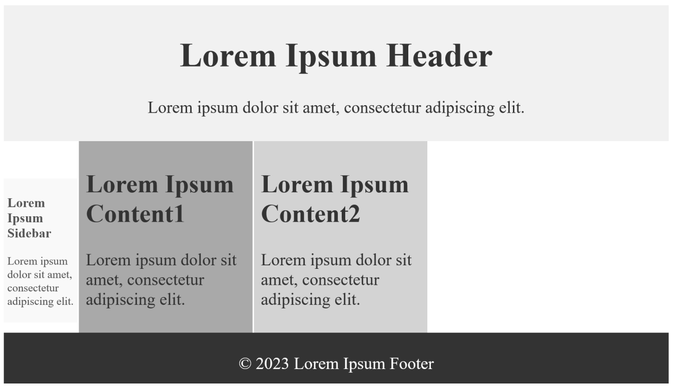


Figure 3. Items rearranged on smaller display

C. Table Layout

As already described in section IV B, HTML-tables used to be widely used for layouts of websites. However, Table Layouts can not only be built using HTML-tables but also with CSS. This option was introduced with CSS 2 [25]. The design technique uses `<div>`-elements in combination with `display: table`-attributes to have a table-like look and feel without infringing upon the structural part of the website which is solely a task for HTML [24, pp. 40-42]. It has its uses for vertical alignment or multiple-column layout with matching heights [25]. But why use this technique and not just a simple HTML-Table? There are two main reasons: First, the strict separation of structure, layout and presentation as described above. The second reason is accessibility: Visually impaired people use screen readers. These rely heavily on the correct usage of HTML-tags like `<table>`. Some people need to use their own style sheets that also rely on the structure of the site, for example people with certain disabilities [22]. The problem with using tables, either CSS or HTML, is that they don't


Lorem Ipsum Header

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem Ipsum Sidebar

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem Ipsum Content1

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem Ipsum Content2

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

© 2023 Lorem Ipsum Footer

Figure 4. Layout using <div>

Listing 3. Example HTML <div>
```
[...]
<body>
  <div id="header">
    <h1>Lorem Ipsum Header</h1>
    <p>Lorem ipsum dolor sit amet, consectetur
        adipiscing elit.</p>
  </div>
  <div id="containerOuter">
    <div id="sidebar">
      <h3>Lorem Ipsum Sidebar</h3>
      <p>Lorem ipsum dolor sit amet, consectetur
           adipiscing elit.</p>
    </div>
    <div id="containerInner">
      <div id="content1">
        <h2>Lorem Ipsum Content1</h2>
        <p>Lorem ipsum dolor sit amet,
            consectetur adipiscing elit.</p>
      </div>
      <div id="content2">
        <h2>Lorem Ipsum Content2</h2>
        <p>Lorem ipsum dolor sit amet,
            consectetur adipiscing elit.</p>
      </div>
    </div>
  </div>
[...]
```

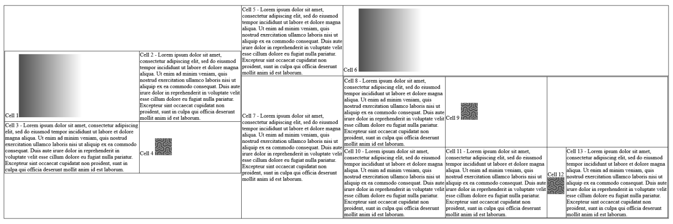


Figure 5. Classic Table Layout

Listing 4. Example CSS <div>
```
#header {
  background-color: #f1f1f1;
  color: #333;
  padding: 20px;
  text-align: center;
}

#footer {
  background-color: #333;
  color: #fff;
  padding: 10px;
  text-align: center;
}
#content1 {
  background-color: darkgrey  ;
  color: #333;
  padding: 20px;
  width: 30%;
  display: inline-block;
}
 [...]
#containerOuter {
  width: 100%;
}
#containerInner {
  width: 80%;
  display: inline-block;
}
 [...]
```

Listing 5. Example HTML-table
```
[...]
<body>
  <table border='1' style=
  'border-collapse:collapse'>
    <tr>
      <td colspan="2" rowspan="2">
        <table border='1' style=
        'border-collapse:collapse'>
          <tr>
            <td>Cell 1
            <img src="examplePic1.png"></td>
            <td>Cell 2 - Lorem ipsum [...]</td>
          </tr>
          <tr>
            <td>Cell 3 - Lorem ipsum [...]/td>
            <td>Cell 4
            <img src="examplePic2.png"></td>
          </tr>
        </table>
      </td>
      <td>Cell 5 - Lorem ipsum [...]</td>
      <td>Cell 6 <img src="examplePic1.png">
      </td>
    </tr>
    [...]
```

handle well in the context of responsiveness. A table has a fixed number of rows and columns. They are also fixed in relation to each other. For responsiveness this means that a table will always be a table. If the screen size decreases there are only two options: Either the table shrinks with the screen size leading to illegibility of the content, or a scrollbar is added. Both options are not desirable in modern web layout design. Figure 5 shows a classic table layout using HTML-tables. As you can see it is rather difficult to generate a decent layout with nested tables. Some cells appear to be empty and the cells do not align properly. The HTML-code for this is listing 5. Additional CSS was added to make the table borders visible.

D. Flexible Box Layout

The origins of *Flexible Box Layout* or *Flexbox*, one of the newer layout techniques fully introduced with CSS 3, date back to before 2008 [26]. A first working draft was proposed in 2009 and a first Candidate Recommendation Snapshot was published in 2012 [27]. This first Recommendation was not immediately supported by all browsers, yet it was deemed practical for implementing responsive designs [28]. With later Recommendations it became such a a success, that it has been one of the most used layout techniques since May 2015 [26]. As of today the *Flexible Box Layout Module* is supported by 99.79% of installed desktop and 100% of mobile browsers [29]. Its intent is not to provide a solution for the complete layout for a website [24, p. 10]. It is rather used for implementing elements for complex user interfaces [24, p. 74]. *Flexbox* utilizes so-called *flex containers*. These containers have child elements that can be arranged around an axis. This axis can *either* be horizontally or vertically aligned. As needed, the child elements grow or shrink to fill the container so it fits the screen. Additionally they do not "leak" out of the container like floats can [24, p. 73]. A layout using *Flexbox* can be seen in figure 7. The containers are defined within <div>-tags in the HTML-code(listing 6). The associated CSS can be found in listing 7.

E. Multi-column Layout

Another module for designing layouts introduced with CSS 3 is the *Multi-column Layout*. The first draft for this module was published in 1999 [30]. The first Candidate Recommendation came out in 2009 [30] and the latest version in 2021 [31]. The name of this technique, *Multi-column Layout*, pretty much speaks for itself. It arranges texts in columns like in a newspaper. This also works in a responsive environment. The columns are automatically filled with text. The number of columns is reduced when displayed on a smaller screen [24, p. 10]. It has great benefits for being displayed on older browsers as there is no need to implement fallback solutions. Fallback is handled automatically, as *Multi-column Layout* extends the existing CSS Box Model [24, p. 43]. It automatically wraps content across columns, and the borders between columns and gutter width can be defined. The content can either expand to fill the container or adhere to a preset size [24, p. 45]. However, it does not have the range of *CSS Grid Layout* because it can only work in columns. An example can be seen in figure8. Note how the different background colors mark the different columns defined in HTML. This is especially noticeable in the middle column. However, the HTML-columns do not coincide with the actual columns shown on screen. The corresponding code can be seen in listings 9 and 8.

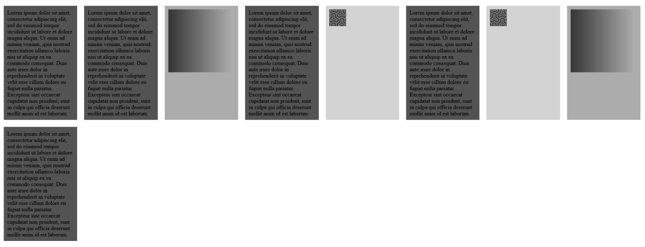


Figure 7. A layout design using *Flexbox*

Listing 6. Example HTML Flexbox
```
<body>
  <div class="container">
    <div class="childText">Lorem ipsum [...]<
    /div>
    [...]
    <div class="childPic1">
    <img src="examplePic1.png"></div>
[...]
```

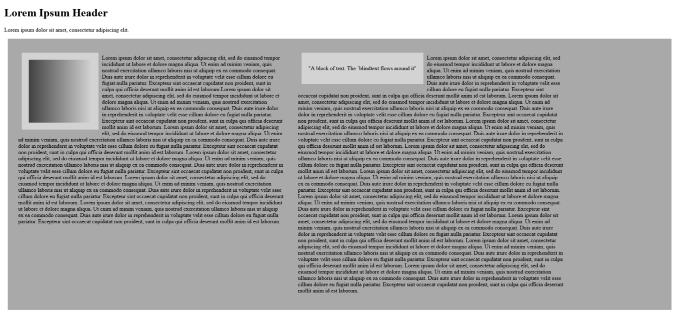

Lorem Ipsum Header
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
"A block of text. The ´blindtext flows around it"

Figure 6. A layout design using floats

Listing 7. Example CSS Flexbox
```
.container {
  display: flex; /* activate the Flexbox Module
      */
  flex-direction: row;
  flex-wrap: wrap;
}
/* all direct children of .container */
.container>div {
  width: 10%;
  height: auto;
  margin: 10px;
  padding: 10px;
}
[...]
```

V. *CSS Grid Layout* benefits

There are many advantages a *CSS Grid Layout* has over older approaches. A website can now truly have a grid layout similar to layouts in print-media. This is a technique that older approaches in web design never quite achieved. This especially applies to responsive websites. To grasp the full potential of *CSS Grid Layout* in comparison to other techniques, it is best to see each technique in action within the same context and application. A direct comparison between the older approaches and *CSS Grid Layout* can be seen in parts 1-5 of figure 9. Every part shows a website created with a technique on a desktop and mobile sized screen, respectively. In the first part a classic *Table Layout* can be seen. As the screen-size shrinks the website does not adapt at all. Instead some scrollbars appear. In the second part a *Floated Layout* can be seen. The right element is infringing on the box of the left element. The third part shows a *Flexbox* layout. It looks good at first glance, but would not really work with horizontal and vertical alignments. *Flexbox* containers can only be aligned on one axis. The disadvantages are very clear here. The content leaks out of their containers and the text gets very small. This is not the case with *CSS Grid Layout*. The fourth part shows a Multi-column layout. The amount of columns is reduced. The last part shows how beautifully *CSS Grid Layout* arranges the individual items on any screen size. The added benefit is, that this would still work if items were added later.

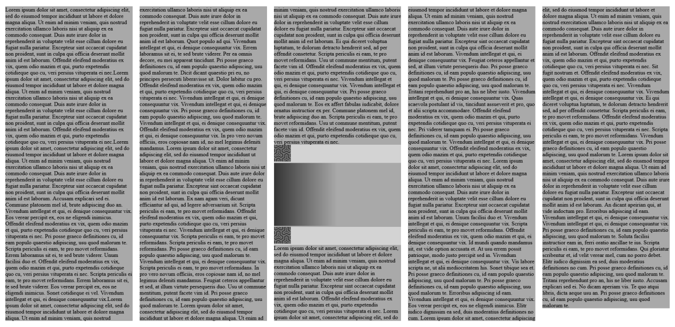


Figure 8. A Multi-column layout

Listing 8. Example HTML Flexbox
```
<body>
  <div class="container">
    <div class="column" id="c1">Lorem ipsum
        [...]</div>
    <div class="column" id="c2">
    <img src="examplePic2.png"></div>
    <div class="column" id="c3">
    <img src="examplePic1.png"></div>
    <div class="column" id="c4">
    <img src="examplePic2.png"></div>
    <div class="column" id="c5">Lorem ipsum
        [...]</div>
</body>
```

Listing 9. Example CSS Flexbox
```
.container {
  column-count: 5; /* number of columns */
  column-gap: 20px; /* gap between columns */
}
.column {
  break-inside: auto; /* prevent column breaks
      within elements */
}
[...]
```

VI. Conclusion

While being around for the better half of the last century, the grid as a tool for organizing and styling layouts in print media is still very relevant today. Its ease of use and intuitive application is easy to learn for beginners in layout design. The flexibility it brings and the way it helps to harmonize the typography of a page, or even a whole series makes it one of the main concepts in use today, even for designers with a lot of experience. However the grid as a concept for good and harmonious layouts is not only relevant to print. There has been a struggle to properly design a layout for a website since the inception of HTML. The goal is to achieve a print-media-like look and feel and thus coherent layouts exhibiting harmony. This has been made even more difficult by the introduction of mobile devices and many different screen sizes. There have been plenty of approaches to solutions for this problem in the past. But every single one of them had its shortcomings. The introduction of *CSS Grid Layout* revolutionized the design of responsive layouts for websites. One could argue that there are still reasons to work with older layout techniques. And they would not be wrong. But the older layout techniques only have a very narrow applicability in very few use cases as we have seen in this article.

References

[1] Apple. Apple Reinvents the Phone with iPhone. (accessed: 07.10.2023). [Online]. Available: https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/

[2] ——. iPad Available in US on April 3. (accessed: 07.10.2023). [Online]. Available: https://www.apple.com/newsroom/2010/03/05iPad-Available-in-US-on-April-3/

[3] P. Anz. Vor 14 Jahren: Erstes Android-Smartphone kommt auf den Markt. (accessed: 07.10.2023). [Online]. Available: https://www.inside-it.ch/vor-14-jahren-erstes-android-smartphone-kommt-auf-den-markt-20220923

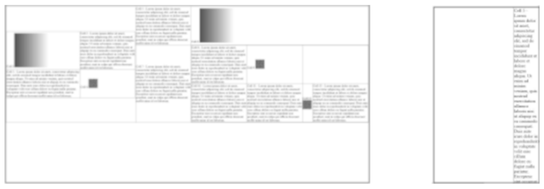
[4] J. Dischler. Buiding for the next moment. (accessed: 07.10.2023). [Online]. Available: https://adwords.googleblog.com/2015/05/building-for-next-moment.html

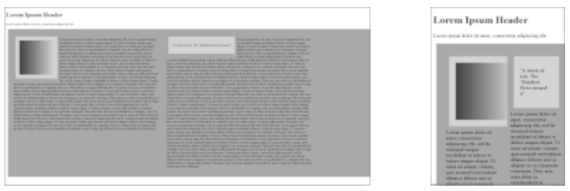
[5] T. Bianchi. Share of global mobile website traffic 2015-2022. (accessed: 07.10.2023). [Online]. Available: https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/

[6] D. Phan. Mobile-first indexing. (accessed: 07.10.2023). [Online]. Available: https://developers.google.com/search/blog/2016/11/mobile-first-indexing

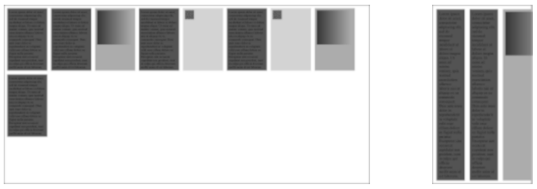
[7] J. Mueller. Announcing mobile first indexing for the whole web . (accessed: 07.10.2023). [Online]. Available: https://developers.google.com/search/blog/2020/03/announcing-mobile-first-indexing-for

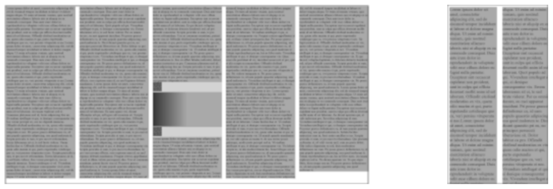
[8] A. Gustafson. The Story of CSS Grid, from Its Creators. (accessed: 07.10.2023). [Online]. Available: https://alistapart.com/article/the-story-of-css-grid-from-its-creators/


1. Layout Using HTML Tables



2. Layout Using Floats


3. Layout Using Flexbox


4. Layout Using Multiple Columns

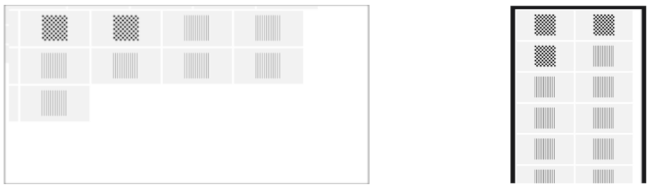

5. Layout Using CSS Grid

Figure 9. Comparison of techniques and their responsiveness

[9] J. Müller-Brockmann, *Grid systems in graphic design : a visual communication manual for graphic designers, typographers and three dimensional designers. = Raster systeme fur die visuelle Gestaltung : Ein Handbuch für Grafiker, Typografen und Ausstellungsgestalter*, ser. 1st edition. A. Niggli, 1981.

[10] C. G. Damien Gautier, *Gestaltung, Typographie etc. - Ein Handbuch*, ser. 2nd edition. Niggli, 2018.

[11] T. Samara, *Making and Breaking the Grid: A Graphic Design Layout Workshop*, ser. 2nd edition. Quarto Publishing Group USA Inc, 2017.

[12] B. Bos. A brief history of CSS until 2016. (accessed: 21.10.2023). [Online]. Available: https://www.w3.org/Style/CSS20/history.html

[13] W3C. Cascading Style Sheets - Levels, snapshots, modules.... (accessed: 22.01.2024). [Online]. Available: https://www.w3.org/Style/2011/CSS-process

[14] ——. CSS Grid Layout Module Level 1 publication history. (accessed: 25.10.2023). [Online]. Available: https://www.w3.org/standards/history/css-grid-1/

[15] ——. CSS Grid Layout Module Level 2 publication history. (accessed: 25.10.2023). [Online]. Available: https://www.w3.org/standards/history/css-grid-2/

[16] ——. CSS Grid Layout Module Level 1: W3C Candidate Recommendation Draft, 18 December 2020. (accessed: 25.10.2023). [Online]. Available: https://www.w3.org/TR/2020/CRD-css-grid-1-20201218/

[17] J. Attardi, *Modern CSS*, ser. 1st edition. Apress, 2020.

[18] B. Frain, *Responsive Web Design with HTML5 and CSS*, ser. 3rd edition. Packt Publishing, 2020.

[19] P. Thormeier, *Mastering CSS Grid*, ser. 1st edition. Packt Publishing, 2023.

[20] T. Berners-Lee, *Weaving the Web: the original design of the Wolrd Wide Web by its inventor*, ser. 1st edition. HarperCollins Publishers, 2000.

[21] NCSA. NCSA Mosaic change logs. (accessed: 21.10.2023). [Online]. Available: https://web.archive.org/web/20081006065314/http://wvnvms.wvnet.edu/vmswww/mosaic/help-on-version-0_13.html

[22] E. Eggert. Tables Tutorial. (accessed: 21.10.2023). [Online]. Available: https://www.w3.org/WAI/tutorials/tables/

[23] W3C. Cascading Style Sheets, level 1. (accessed: 21.10.2023). [Online]. Available: https://www.w3.org/TR/REC-CSS1/

[24] S. Hampton-Smith, *Pro CSS3 Layout Techniques: Leverage the power of CSS3 to create sophisticated layouts*, ser. 1st edition. Apress, 2016.

[25] W3C. Tables. (accessed: 21.10.2023). [Online]. Available: https://www.w3.org/TR/CSS2/tables.html

[26] M. Stoeva, "Evolution of Website Layout Techniques," in *Anniversary International Scientific Conference "Computer Technologies and Applications", 15-17 September 2021, Pamporovo, Bulgaria*, 2021.

[27] W3C. CSS Flexible Box Layout Module Level 1 publication history. (accessed: 22.10.2023). [Online]. Available: https://www.w3.org/standards/history/css-flexbox-1/

[28] F. Maurice, *CSS3:Die neuen Features für fortgeschrittene Webdesigner*, ser. 2nd edition. dpunkt.verlag, 2014.

[29] Caniuse.com. CSS Flexible Box Layout Module. (accessed: 22.10.2023). [Online]. Available: https://caniuse.com/flexbox

[30] W3C. Multi-column layout in CSS. (accessed: 09.12.2023). [Online]. Available: https://www.w3.org/1999/06/WD-css3-multicol-19990623

[31] ——. CSS Multi-column Layout Module Level 1. (accessed: 09.12.2023). [Online]. Available: https://www.w3.org/TR/css-multicol-1/