University of Tripoli
Faculty of Sciences
Department of Computer sciences

Responsive Web Application Design

Mohamed Ahmed Amhimmid Elawer

Mustafa Fateh Abdulaal
Professor

A Thesis Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of science in Computer Science

24/10/2023

DECLARATION

I Mohamed Ahmed Elawer the undersigned hereby confirm that the work contained in this thesis, unless otherwise referenced is the researcher's work, and has not been previously submitted to meet the requirements of an award at this university or any other higher education or research institution, I furthermore, cede copyright of this thesis in favor of the University of Tripoli.

Name: Mohamed Ahmed Elawer

Signature: ....

Date: 24/ 10/ 2023

ABSTRACT

Responsive web design (RWD) is a new approach to web development that automatically adapts itself to different browsers and screen sizes of users whether it is a large desktop screen or a small smartphone.

This thesis surveys the popular RWD software technologies and tools extensively. Unlike other surveys, this survey differentiates between software tools and software technologies used in RWD. It is very clear from the survey that the current state of the art comes down to two most effective RWD technologies namely the CSS Grid and the CSS Flexbox [1, 2, 3]. Therefore the thesis provides deep analyses and comparisons of these two technologies showing their flaws and cons.

Based on these comparisons the thesis proposes a new RWD model named RWAD (Responsive Web Application Design) that combines the above two mentioned technologies by adopting their advantages and discarding their disadvantages. Since the implementation method is the main key for the success of any RWD technologies [4], the proposed RWAD model is formulated algorithmically in an implementation mode using a full-fledged RWD site with diversified content so that it can be used as a guideline model.

Combining these two technologies has been done before, especially in the well-known package of Bootstrap (latest version 5) [5,6], but it is very difficult, if not impossible, in Bootstrap to correct any errors unless restarting from the beginning and reloading huge amount of library codes [7, 8].

In RWAD, the CSS Grid and the CSS Flexbox techniques are integrated without using Bootstrap or any other external code, and with the ability to correct, maintain, and update the RWD site directly without restarting the whole process or reloading any external codes. This of course saves time and space and provides an efficient implementation of responsive sites.

ACKNOWLEDGEMENTS

To the one who led the hearts and minds of humanity to the port of safety, the first teacher of humanity (Muhammad peace be upon him),

To the one who left me with her body but is still present in my memory..... My dear mother, may God have mercy on her soul,

To the one who was my support, condolences, strength, and glory..... My dear father,

To the one who was my shadow when fatigue enveloped me, and she is the one who supported me and helped me ... My dear wife,

To the seeds of my heart, and my hope of tomorrow …. My beloved children (Roa'a, Rahaf, Abdulrahman),

To all my teachers especially those in the department of computer sciences of the University of Tripoli, and especially he who was behind the success of this work and was the best help for me in my career..... Pr. Dr. Mustafa Abdulaal,

I dedicate this thesis and hope to God that it will be a useful piece of knowledge and an additional knowledge card...

Mohamed Ahmed Elawer

Tripoli, 21/05/2023

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURES

LIST OF ABBREVIATIONS

RWD	Responsive Web Design
HTML5	Hypertext Markup Language
CSS	Cascading Style Sheets
Flexbox	Flexible Box Module
CSS Grid	Grid-based layout system with rows and columns
Px	Pixel
UI	User Interface
SVG	Scalable Vector Graphics
JPEG	Joint Photographic Experts Group
Div	division element
RF	Edge Reflow
MIX	Maximum
MIN	Minimum
D	Dimensions
App	application
IN	Adobe Edge Inspect
IE	Internet Explorer
WWW	World Wide Web

1 INTRODUCTION

Websites are no longer limited to displaying attractive content but must deal with many other issues. One of the prominent issues in web development is responsive design so that customers can view the same content regardless of their devices of different screen sizes [9]. Many websites are designed particularly for computer screens. They soon become unfriendly and lose content when used by smartphones or iPads. Of course, the design of a version of the same website for each screen size is costly and time-consuming. Adaptive methodologies did not solve the problem, as it is a series of fixed-width layouts. Responsive Web Design (RWD) on the other hand provides flexible dimensions making the web page self-adjustable on any screen size without any loss of content. With responsive web design, one can make sure that the website looks its best on cell phones, tablets, laptops, and desktop screens [10]**.**

RWD is now a necessity for the following reasons:

- The proportion of mobile Internet users is about 60% of the total use of the Internet**.**

  The percentage of global web traffic on mobile phones has surged over the past decade. In February 2023, 60.67% of all web traffic came through mobile phones, with the average for 2023 so far coming in at 60.06% [11].

  If we were to go back to 2012, this figure was at a meager 10.88%. Fast forward five years later to 2017, and the percentage of web traffic on mobile had already increased five-fold to 54.09%[12].

  The quickest growth rate came from the first few years of the past decade and has slowed significantly since then. Comparing the percentage of web traffic on mobile, the average year-over-year growth rate from 2013 to 2018 was 32%, compared to 2% from 2018 to 2023[13].

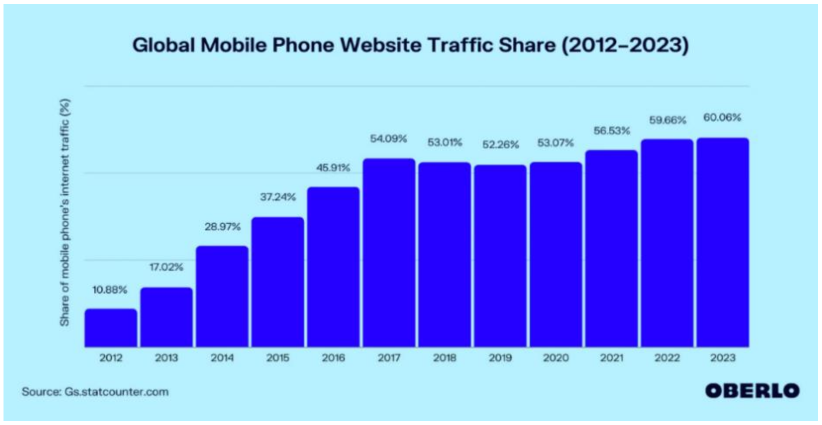


Figure 1: Percentage of Global Mobile Traffic (2012-2023).

- Google presently ranks web applications based on the design response to different users.

  Devices, hence penalizing sites that do not respond to mobile phones. Mobile-friendly is a proven ranking factor for Google. If a webpage shows less information on mobile than on desktop, Google won't consider it for rankings. This is fed right into the Google Page Experience Algorithm Refresh, which was introduced in 2020. That is, websites are at a disadvantage for Google if it is not mobile-compatible. Businesses with outdated website designs should strongly consider upgrading to remain competitive at Google [14].

1.1 The Problem Statement

Many research efforts are still going on to find effective ways to implement RWD as can be seen from the literature survey in Chapter 2. This thesis is part of these efforts, starting by surveying the available RWD software technologies and the available tools.

It is apparent from the literature that the focus lies on two state-of-art RWD dominant technologies namely CSS Flexbox and CSS Grid. This thesis performs a deep analysis of both of them, identifies with practical verifications their differences and similarities, determine when and how to select any of them, and shows how they can be best implemented.

Then the thesis provides a new model combining both of them (RWAD) by considering their advantages and discarding their disadvantages.

As the implementation method is an important factor in ensuring a successful RWD site [25], a full-fledged RWD site with diversified content is developed and implemented using the proposed RWAD model.

This combination has been done before, notably in the latest version of the well-known bootstrap5 package [5]. However, in bootstrap integration, it is not possible to determine or correct any error without restarting from scratch and reloading a huge amount of unnecessary codes. The combined model proposed in this thesis does not suffer from this drawback, as it is possible to correct and update the site without restarting or reloading any codes. Therefore, saving implementation time and space.

1.2 Research objective

The objectives of the thesis can be outlined as follows:

- RWD is best achieved by applying an integrated set of techniques with appropriate tools. Unfortunately, no definite evaluations or clear indications to guide developers on how to select and implement appropriate technologies for their web applications. This thesis provides clear guidelines to help developers design and implement their RWD sites. Moreover, The proposed RWAD provides a guideline model to efficiently build an RWD application site.

- RWD tools and techniques are complicated to learn and implement, especially for inexperienced web developers. This thesis simplifies the learning process by classifying layout planning methods, distinguishing design technologies from software tools, and comparing the main characteristics of modern effective methods theoretically and practically.

- RWD is not notably used in Libya, nor is it part of any curriculum in our higher education, while responsive design is increasingly important. This thesis would hopefully promote RWD.

1.3 Page Layout Planning

Page layout is the main part of web design because it determines the arrangement of elements on the web page. This is a very complicated matter if the web page is to be displayed in different browsers and different screen sizes (RWD). This thesis especially addresses the differences resulting from different screen sizes. The following sections briefly outline the most common layout methods before and after the era of RWD.

1.4 Layout Variations

There are three well-known approaches to treating layout variations:

1.4.1 Fixed-Width Layouts

Fixed layout, also known as static layout, is based on fixed width measured in pixels. So no matter what the screen size or resolution, the width of the elements will remain the same. In other words, the viewing experience of the users will remain the same across all device types be it computer, laptop, smartphone, or tablet. [15].

The “**Float**” technique is used to implement a fixed layout because of its convenient properties to manipulate columns without using tables, as is the case in HTML. Figure 2 is an output of a fixed layout as it appears on the desktop screen and a mobile screen.

Disadvantages: Fixed layout becomes obsolete because it creates a lot of blank space for screens of high resolution, and causes the appearance of a horizontal scrollbar for screen resolution of low [14].

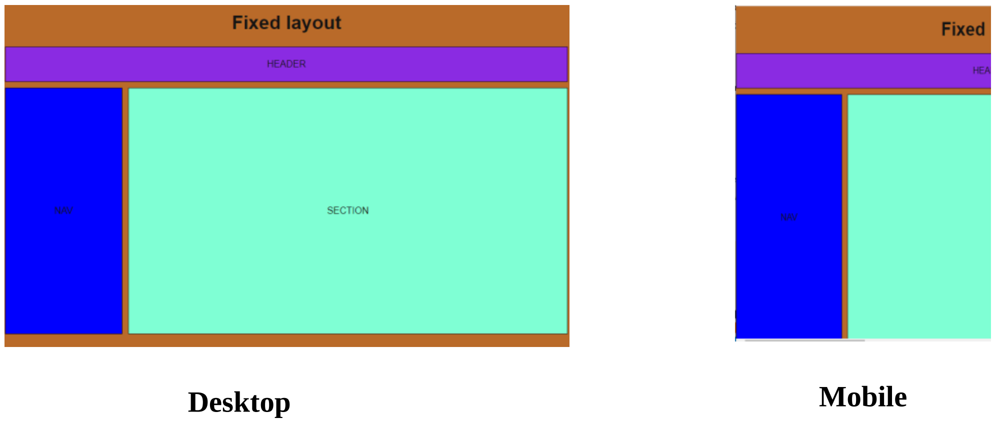


Figure 2: It is the product of an implementing program for the Fixed planning policy

1.4.2 Fluid layout

In a fluid layout, components are adjusted according to the screen size of the viewing device. The width is measured in percentages instead of pixels. The benefit is that when screen size changes, for instance from a laptop to a smartphone, the proportion of elements remains unchanged. The content can expand or shrink to fit the window of the user's computer [15]. The fluid layout is also implemented by the “**Float**” technique and the output is shown in Figure 3

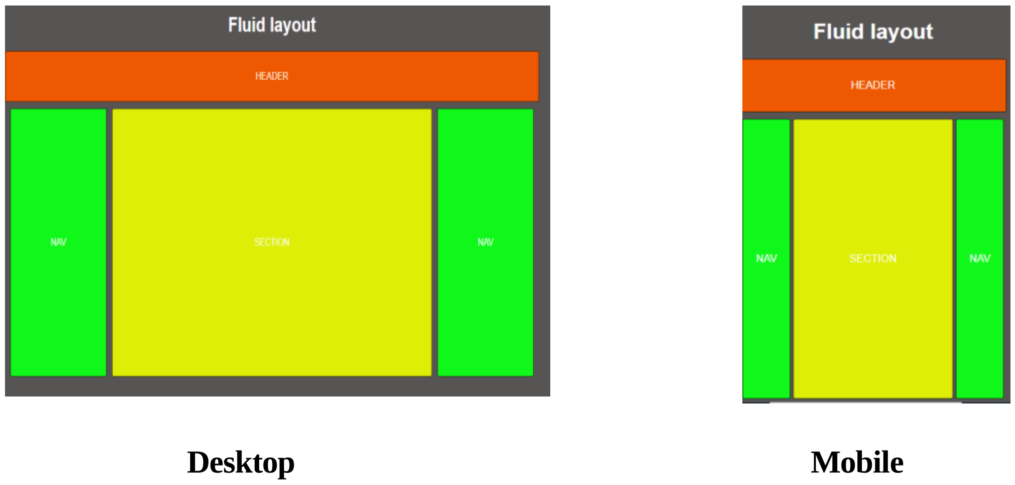
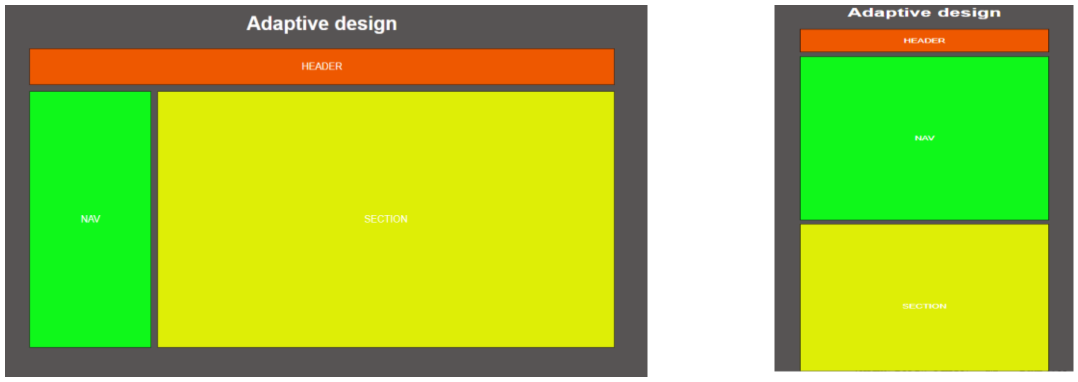


Figure 3 It is the product of an implementing program for the Fluid planning policy

Disadvantages: It is almost impossible to know how it will look on different resolutions. Large screen resolution creates long unreadable paragraph lines and/or a lot of free space. Graphical content elements must have multiple-width properties to accommodate different screen resolutions [16].

1.4.3 Adaptive layout

The adaptive layout or adaptive design refers to a graphical user interface (GUI) design that adapts to different screen sizes. The content follows a fixed layout size by developing designs for the most common screen widths, e.g. six sizes, 320, 480, 760, 960, 1200, and 1600 pixels which are standard practice for designers. This is the first time the "**Media queries"** technique has been used to implement page layout to allow changes according to the screen sizes, which makes page content feasible in all of the predetermined screen sizes [17]. The output generated from our adaptive layout code is presented in Figure 4.



DESKTOP **MOBILE**

Figure 4 It is the product of an implementing program for the Adaptive planning policy

The **advantage** of adaptive design is that it allows the designer to tailor-make solutions that appear optimal on different screen sizes.

Disadvantage: It requires the creation of multiple designs (six or a nutshell equivalent to six versions of a single webpage) to have the best one ready for the needed screen specifications. Another disadvantage is that adaptive design might leave users who don't have a standard-sized screen without an optimal solution. Nevertheless, it remains one of the options not to be dismissed [16]

The fixed layout does not take into account the sizes of different screens. It is divided by pixels and produces empty white spaces on small screens. An unwanted scroll bar appears in the design. The flexible layout takes into account the sizes of different screens and is divided by percentages, but it does not look at the internal elements of the blocks, one can not expect what happens on smaller screens. The adaptive layout method is just a collection of versions for different screen sizes, i.e. it is a set of fixed layouts. The problem with the adaptive layout method is that it takes time to issue more than one layout, and if there is another different screen size, there will be no suitable layout for it.

1.5 Treatments of Layout Variations

The following methods are used to treat the problem of layout variations. Some methods are becoming out of date, some are not sufficient to work alone, and the latest ones are modern and most effective. They are all called Layout Planning Methods.

1.5.1 Legacy Layout Methods

These are relatively old but were popular layout planning techniques such as CSS Position and Floats:

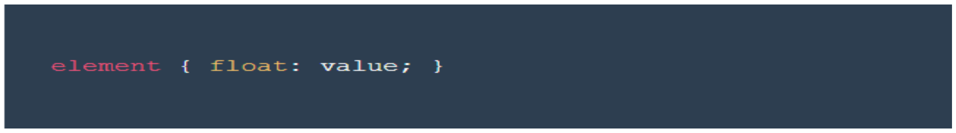
I. Position

The CSS position property is used to modify the position of any element in the direction the element is required to move such as up, down, right, or left [18].

II. Floats

Property is, generally speaking, to push a block-level element to the left or right, taking it out of the flow of other block elements [19].

Syntax: The float property in CSS is defined inside the brackets as shown, where the Value is to be specified as Left, Right, or None



```
element { float: value; }
```

The CSS float property works like an HTML table but it can flexibly accommodate various layouts without any changes in the code.

Both Position and float are relatively old CSS methods to customize the layout of pages, with limited properties. They are considered non-responsive and referred to as the Legacy Layout Method. This does not mean that these technologies are no longer used, but rather are used with modern technologies now.

For Example:the famous RWD software Bootstrap has used CSS Flout in its old versions including version 3.

1.5.2 Basic Layout Methods

Much more effective methods than the legacy ones were needed. The two most basic ones have emerged and are presented here**:**

I. The Flexible grid concept

"Flexible Grid" calls for page element sizing to be in relative rather than fixed units, as described in the fluid layout strategy above. The main idea is to create a layout where all elements are based on the calculated percentage width and so all elements in the layout are resizable about one another.

To calculate an element's proportions, designers have to take the width of the element and divide it by the size of the parent element, for example, 200 px / 960 px = 0.2083, and multiply the result by 100 to get the percentage value to perform a correct resizing 0.2083 * 100 = 20,83% as shown in Figure 5.

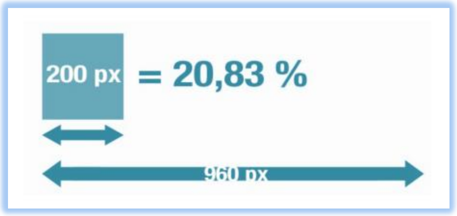


Figure 5: Demonstrates an example Flexible grid concept

Flexible grid design considers layout width, minimal and maximal width, and takes care that the design is not too large when the screen resolutions increase [20].

II. Flexible images

Flexible images are similar to a "Flexible grid" but applied to image elements**.** It allows the image to grow and shrink in size with your screen**.** Flexible images will even generate smaller-resolution images to speed up mobile browsing. The following figure is a result of a simple code to check the method [21].

With the measurement in %, the image can resize itself as its container resizes. If an image is larger than its container it will be scaled back to be the same size as the container. If the image is smaller than the container, the image will be its default size.

The max-width property keeps elements from growing large with one line of code:-

```
Img {
 max-width: 100%;
 }
```

Mobile **Desktop**

Figure 6: Shows Different sizes of the browser window to show how the image will be resized

III. Media Queries

Media queries is a CSS3 module allowing content rendering to adapt to conditions such as screen resolution (e.g. smartphone screen vs. computer screen). It is a cornerstone of RWD technology. Among the features that can be used in media queries are "width", "height", "orientation" and "color" **[22].**

Syntax: A media query is composed of an optional media type and any number of media feature expressions. Multiple queries can be combined in various ways by using logical operators.

The general form of a media query is shown in the code:

1. 1 . @media [not|only] type [and] (expr) {
2. /* CSS rules used when query matches */
3. rules
4. }

Example: If the browser window is 550 pixels or smaller, the background color will be yellow:

```
body

  background-color: light green;

@media only screen and (max-width: 550px)

{

  body {    background-color: rgb(249, 253, 13);

}
```

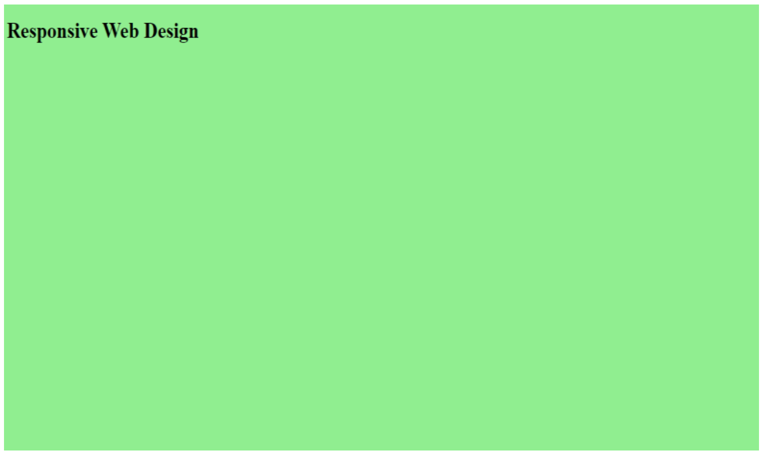

Responsive Web Design

Figure 7: Display results in case of large screens

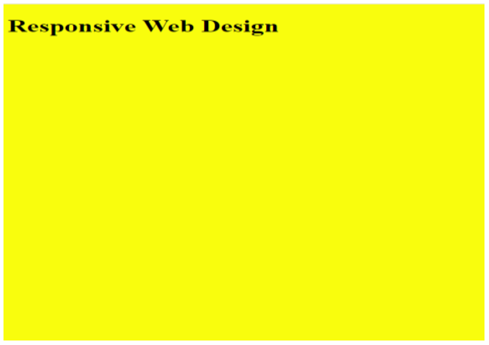

Responsive Web Design

Figure 8: Display results in the case of screens less than 550 pixels, i.e. small screens

1.5.3 Modern Layout Methods

All of the website layout options described above have a lot of shortcomings that provoked a new approach for more effective RWD techniques. Web development has reached a point where two CSS layout systems are dominantly effective [23, 24]:

CSS flexbox and CSS grid

CSS Flexbox and Grid are responsive layout modules with very powerful CSS page planning tools that have become prevalent in recent years. They are widely supported by platforms such as browsers and libraries. A deep study and comparison of these two modules will be introduced in Chapter 3. These two technologies have characteristics with values that can be updated to make them responsive to cases with all types of screen sizes. It is not like the previous float and other techniques, which lack characteristics' values, hence, letting clashes occur when used in interactive sites with different screen sizes.

1.6 Drawbacks and Limitations of RWD

- **Arranging the elements: -**

One of the most important problems raised in this study is how to arrange the elements based on the screen size. This problem is dealt with by the techniques studied in this thesis namely: grid, and flexbox.

- **Moving:-**

Of course, the user on large screens has the flexibility to move between the site's buttons and move from one page to another. During the design, one must give the same flexibility to make the navigation intuitive and clear on all screens.

- **View large tables:-**

Displaying spreadsheets (flight schedules, for example) on small screens is a real problem when the tables are complex and large, the user usually prefers to use these tables on the computer, but to be responsive, one must enable the user to get table information on the phone screen.

- **Converting of fixed websites:-**

Converting fixed sites into responsive sites has bigger problems because the developer who designed the site on computer screens did not think about it.

1.7 The Thesis Methodology

- Review the RWD through two important surveys**:**
  i. Survey of the relevant literature of previous research on RWD technologies with emphasis on technologies used in web planning.
  ii. Survey the design tools commonly used with any RWD technologies.

- An extensive study of the currently recognized technologies (CSS grid and CSS flexbox) with theoretical and practical comparisons leads to how to select and best implement each of them.

- Propose an effective model to combine the two technologies adopting their advantages and avoiding their disadvantages.

- Practical design and implementation of a responsive website model (template-like model) using the proposed combined model of the two technologies.

- Conclusion and recommendations.

1.8 Organization of the Thesis

This thesis is organized as follows:

- **Chapter 1** is an introduction to provide a background of the problem under study. It also defines the problem statement, study methodology, and research objectives, and explains the importance of the study.

- **Chapter 2** is a literature review of RWD Technologies in chronological order from the outset of RWD to date. Also, there have been many tools used to support Responsive Web Design (RWD) which are presented separately in this chapter.

- **Chapter 3** is a detailed study with illustrative examples, analysis, and comparison of modern techniques used in RWD, namely CSS flexbox and grid. This includes the design and implementation of a complete model for each technology (flexbox and grid). Therefore, the strengths and weaknesses of each of them are shown practically.

- **Chapter 4** presents a new model to effectively design and implement a combined RWD site by integrating CSS flexbox and CSS grid technologies in one advantageous model with practical illustrations.

- **Chapter 5** is a Conclusion

2 LITERATURE REVIEW

There have been many software **technologies** used in Responsive Web Design (RWD) supported by varieties of software **tools.** Technologies represent the methods and know-how of RWD while tools provide the means to implement these methods. It is like improving or maintaining a device, one must have the know-how to do so but needs tools to execute it. Developers may use different tools to implement different parts of their methods. Selecting the appropriate tools is usually a matter of convenience, availability, and effectiveness.

This thesis is about technologies, not tools even though some tools are used in practical implementations. Tools are always needed to implement any technology no matter what. Therefore, a literature review of RWD technologies is presented and then followed by a separate review of the most available and popular software tools. The distinction between software technologies and software tools is not apparent in the literature.

2.1 Literature review of the RWD software technologies

The following are techniques that have already been studied in previous research:-

- **In 2010 Ethan Marcotte published a book on the RWD approach [26]**

  He coined the term RWD on his website "A Book APART", uniting three existing technologies (flexible grid layout, flexible images, and media queries) into one unified approach, and called the term responsive web design, the idea here is to prevent the user from zooming in and out or scrolling up and down repeatedly to search for the required information. By doing so, the web design thinking has changed to the RWD approach, the last update of the book took place in 2017 [27].

- **In 2011 Eva Harb, Paul Cappelari, Stephen Long, and Norbert Spot surveyed (Responsive Web Design) [20]**

  They identified what responsive design is, as an alternative to static design, pointing to the problem of the huge amount of different screen sizes, and what can be done to solve this problem. This survey was written by them after evaluating about 400 device models collected during the years from 2005 to 2008, they provide an extensive summary of the latest responsive web design with all its different technologies and show some good examples of how to design responsive modern websites, they suggested: desktop philosophy first, mobile philosophy first. They also described and gave examples of the main approaches to responsive web design namely: flexible network, flexible images, media queries, responsive printing, and responsive email systems.

- **In 2012, Ben Frain published a book titled (Responsive Web Design with HTML5 and CSS3) [28]**

  This book offers the complete "how-to" for taking an existing fixed-width design and making it responsive, it provides examples of responsive design techniques using the media queries approach, it also explains fluid layouts and shows how to convert an existing fixed-width design into a fluid layout or use the CSS framework to quickly prototype a responsive design.

- **In the year 2013 Kailashkumar V. Natda [29]**

  His paper shows how to implement techniques in the field of web design.   He also described the experience across a wide range of devices such as desktops, laptops, tablets, smartphones, and other gadgets, using a variety of techniques: Fluid grid layout, Breakpoint, Media Queries, Viewport Responsive Graphics, and other media.

- **In 2014 Thoriq Firdaus published a book [30]**

  Explain how to build a website using HTML and CSS3 with responsive design techniques that work across a variety of devices, the book has some additional features in CSS3 such as transformations and transitions, and explains the power of media queries and the use of flexible layouts and frameworks in responsive design, in addition, how to test a responsive site is also explained.

- **In the year 2014, both Abdulrahman A. Mohamed, Dr. Cheruiyot W.K, Ph.D., Dr. Richard Rimiru, Ph.D., & Collins Ondago surely [31]**

  They presented a chronological in-depth survey and illustrated a shift from traditional web design toward responsive web design, the paper includes some of the modern technical aspects of responsive web design and calls for alternative approaches, techniques covered in this paper include CSS @media queries, Fluid images, and video, JavaScript, often triggered by window match Media, Server-side solutions, and Scalable Vector Graphics (SVG) to create resolution-free images.

- **In 2015 Rogatnev Nikita presented a thesis [16]**

  In this thesis, almost everything related to responsive web design was described with a comparison of all possible site-planning methods such as static or fixed width, fluid, and adapted layout design, their drawbacks are clearly shown so a responsive approach is undoubtedly preferable.

- **In 2017, Luminita GIURGIU, Ilie GLIGOREA presented a paper [25]**

  Explaining that there is no universal solution suitable for anything, and improper implementation can also be a disadvantage, they mention that the current major trend in web design is to start the design from the screen of mobile devices but take into account how they will be viewed on desktop screens, techniques covered in this paper: Flexible Grid-based layout, Flexible media, Media Queries (@media).

- **In 2017, Fernando Almeida, José Monteiro presented a paper [32]**

  To explore the key benefits and limitations associated with responsive web design, they adopted a quantitative approach based on a questionnaire filled out by 181 industry professionals that allowed them to identify reasons why developers adopted the responsive design as well as address the limitations they felt, the results obtained indicate that providing a good user experience and increasing accessibility stand out as the most important advantages, On the other hand, the main limitations include compatibility with older web browsers, increased load time, and difficulties in improving the user experience, Finally, they found that visualizing the advantages and limitations of responsive design differs between professionals with professional experience in the field and independent developers.

- **In 2017, Emmanuel Ohans published a book [33]**

  Demonstrate responsive design using Flexbox technology. He described how to use the properties of the flexible container and align the flexible elements, it describes the understanding of absolute and relative elasticity, automatic margin alignment, flexible direction switching, and how elements are responsive.

- **In 2018, Alireza, Reza Ghazanfar, and Mohsen Reda published a research paper [34]**

  In this paper, a comparison of different types of responsive methods has been done, RWD technologies and tools as well as RWD versioning and revision are discussed in this paper, the paper aims to find the best approach and suggestion for future work, This paper is a guideline for developers who will gain better experience and improve their productivity in this field.

- **In 2018 Ivan Bradač published a research paper[35]**

  This paper presents the capabilities of a newly developed tool for web design and layout of CSS grids, It calls for the need and support from the web developer community for the creation of the grid implementation tool and methods, and it describes what is required to handle CSS Grid and how to use them properly, By introducing the capabilities of CSS Grid and comparing it to CSS Flexbox and Bootstrap, I show why it can be considered a major shift in web design.

- **In 2019, Georg Niess, Arwin Roubal, Stefan Thurner, and Enrique Barba Roque published a paper [36]**

  This paper aims to introduce a state-of-the-art way to layout web pages in CSS, introducing CSS Grid, and its most important principles, properties, features, and options, showcasing why Grid is recommended to use compared to older techniques, Furthermore, this paper explores Grid Generators, including tools available online, and evaluates their usefulness, Moreover, this paper introduces the latest browser tools to examine the grid in the browser source inspectors, That is, this paper gives an overview of a new topic in web page layout design with CSS.

- **In 2019, Suman Aryal's review Paper on the Bootstrap Framework [37]**

  The grid system is the most common feature of Bootstrap, which removes the trouble while designing responsive page layouts, a **grid** layout is about the structure of rows and columns for shaping the contents of web pages, The grid layout manages the horizontal and vertical locating of the webpage contents and the structure of the page on various devices' screens. The columns are rearranged based on the devices, The **Bootstrap** grid system is a **flexbox** with 12 columns to design the layout of various shapes and sizes.

- **In 2020, Juha Selonen did a research paper [38]**

  This study examined the most widely used CSS framework for creating websites and the most widely used among developers, the most popular CSS framework is Bootstrap, which is the most widely used framework for creating web pages, Bootstrap is built using Flexbox and Grid technologies, The page is divided in one-way mode and the content is displayed on the page according to the Flexbox specification, It uses another responsive design technology, CSS Grid, in page layout. It is newer and is based on a grid that works in portrait and landscape orientation.

- **In 2020 P. Ajitha published a paper [39].**

  This paper explains how responsive design allows software developers to create a web page that can dynamically adapt to the size of screen devices, ensuring a good user experience on mobile, tablet, and desktop devices, as the world is highly vulnerable to security attacks, the paper mainly focuses on the web application layer and provides a complete survey of current research results on web application security, in this study, he explained how to design a responsive site, but his study was with old techniques and introduced the design of how to protect the site through algorithms, and these algorithms have nothing to do with responsive design techniques.

- **In 2020, Yu Zhang in his thesis [40]**

  In this study, he presented the basic techniques for web user interface development (HTML, CSS, and JavaScript), These technologies are closely related to the C-RWD development process, Then, from the historical perspective of web layout design, he reviews the evolution of web layout design and eventually moves on to responsive web design, which he introduced as a core topic of study, He introduced computational methods from the perspective of user interface improvement and explained the role and application of computing methods in interface improvement, proposed a model to implement RWD and called it C-RWD (Responsive Web Design Service), which provides services on the LaaS platform. The contribution of C-RWD is that it can reduce the time and cost of RWD production and generate a personalized and optimized interface for users. This study touched on web development techniques and defined HTML, CSS, and JavaScript, as a work environment not as techniques, This thesis explains the difference between the approach and techniques used by the approach.

- **In the year 2021, Maya Stoeva published a paper [41]**

  This paper provides a look at RWD technologies with a brief overview of the evolution of the most important milestones of vital technologies in the web from inception to the present, this paper aims to reveal the evolution of web layout technologies over the years and to illustrate some of their future perspectives. This study has details about previous and current techniques, although it suffers from confusion among techniques, tools, and the work environment.

  In the year 2022, Sagar Regmi published a paper [42]

  This paper is based on a dissertation on the concept of creating a responsive web design and implementing it to build a website optimized for responsive design, the theoretical part of this work explains how to create an adaptive design, including fluid text, fluid grid, and adaptive media, as well as media queries, for practical purposes, a website was created as a prototype, tested on devices with different screen resolutions, and evaluated in terms of adaptive design.

- **In the year 2022, Nikolay Chochev published a paper [1]**

  This paper pays attention to current trends in which the overall look of adaptive web pages is designed through the CSS approaches: grid view (Grid Layout), view from flexible areas (Flexbox Layout), and multiple-column layout, the paper analyzes some of the popular practices in designing modern adaptive web design which is important for quality search engine optimization (SEO - Search Engine Optimization), improving performance, and reducing cost, An overview of modern concepts for designing pages Grid Layout, Flexbox.

The article discusses the main changes that occurred in the grid views of web pages after the advent of HTML5. An overview of modern concepts for designing pages Grid Layout, FlexBox Layout, and Multiple Column Layout was analyzed, showing some of their advantages. It is very clear from the above review that modern web design has to be responsive and the most responsive techniques under consideration nowadays are the CSS Grid and Flexbox, it is also clear that the implementation method is the key to the success of any responsive model. Therefore, this thesis provides a detailed analysis of these two important techniques and practically deduces an efficient method to implement each of them. Then, a combined model (namely RWAD) is proposed with algorithmic implementation.

2.2 Literature Review of the RWD Tools

As mentioned at the beginning of this chapter, tools must be used to implement any technology no matter what. Some tools support the implementation of the responsive design process. Some tools are used to help beginners to use responsive templates, and some tools are for more experienced designers, there are also responsive testing tools to help ensure that the final website works well on all types of devices.

- **Bootstrap**

Bootstrap is a free front-end framework for faster and easier web development Bootstrap includes HTML and CSS-based design templates for typography, forms, buttons, tables, navigation, modals, image carousels, and many others, as well as optional JavaScript plugins Bootstrap also gives you the ability to create responsive designs.[43]

Version: The current version of Bootstrap is Bootstrap 5 – version 5.2, Layout Bootstrap uses Grid and Flexbox for element display options. Breakpoint is used to represent the screen size and viewport dimension.[5.6]

- **Webflow**

It’s a platform or a block of software tools that integrates Flexbox design into a visual builder, saves hours of development, and provides an easy UI to build responsive layouts. It includes tools such as a content management system and a hosting platform, these tools are usually separate from each other but Webflow has combined these tools into one structure [44].

- **Adobe Edge Inspect**

Adobe Edge Inspect (previously known as Adobe Shadow) is a preview and inspection tool that is used to develop and test web projects targeting mobile web browsers, Adobe Edge Inspect, can pair multiple mobile devices and browse in sync with a computer, remotely inspect and debug mobile web projects, and instantly see the changes in the targeted mobile devices, this ensures that mobile web page looks as intended across multiple mobile devices [45], Moreover, the integration between Dreamweaver and Edge Inspect allows to preview and check web pages on multiple devices at the same time using Google Chrome [46].

- **Adaptive images**

  Adaptive Images detect the screen size and automatically deliver device-appropriately re-scaled versions of the web page's embedded HTML images, it is intended for use with Responsive Designs and to be combined with Fluid Image techniques. [47].

- **Mobile-Friendly Test (by Google)**

  Google's Mobile-Friendly Test is not specifically for testing responsive websites, but Google has been upfront for some time about how highly its algorithm considers mobile-friendliness in websites in determining rankings, For any business that cares about SEO (Search Engine Optimization.), Google's tool verifies that a website is mobile-friendly enough to meet their standards [48], The tools check for mobile usability problems such as small font sizes (which are hard to read on a small screen) and the use of Flash (which is not supported by most mobile devices) [49].

- **FitVids**

  A lightweight, easy-to-use jQuery plugin for fluid-width video embeds. Media (files, photos, music, videos, etc,) is what slows the web down the most. Chris Coyier, who runs CSS-Tricks, decided to provide developers with FitVids — a fluid jQuery library for doing seamless video embedding in responsive websites. That way, users may never question the quality of the platform they're browsing[50].

- **FitText**

  FitText makes font-sizes flexible. Use this plugin on your fluid or responsive layout to achieve scalable headlines that fill the width of a parent element.FitText or Text or typography plays a huge role in getting the layout right with aligning typography. This will reflect the same qualities over all devices and mediums that access your content. Similar to FitVids, FitText is a simple jQuery library for scaling headlines that will match the required size. This way, headlines will always remain in the spotlight, fully aligned through modern standards of web design[51].

➢ **CrossBrowserTesting**

RWD focuses not only on different device types and sizes but also on different browsers as another important consideration. The Cross BrowserTesting tool can spot potential issues in how your website will look and work in all the main browsers before they affect visitors' experience of the website[52].

➢ **Golden grid system**

The golden grid system was created by Joni Korpi and goldengridsystem.com and is now owned by Charles Brian International. Golden Grid System is a folding grid system for a responsive structure: It is a 16-column responsive grid designed to be folded into 8 and then 4 columns as the viewport decreases. The system also contained a set of typographic presets so that all font dimensions, margins, paddings, and other measurements were established and effectively aligned. Hence, the site would retain a great vertical rhythm. Because most of these measurements were being established in ems (em and rem are both scalable units that also specify values of properties. em and rem meet web accessibility standards, and, unlike px, scale better. Consequently, they are more suited for responsive design). Every component on the web page might be zoomed in or out to raise health inside the fluid-width columns as needed[53].

➢ **UXPin**

UXPin is useful throughout the whole product creation process as it can create wireframes, UI designs, and interactive prototypes that feel like the real product without writing any code whatsoever. The website owner can understand the ideas of the developer better and collaborate on iterations using this one tool, which makes it effective in developing a real product. UXPIN is a collaborative design and prototyping tool created to reduce the time spent on design and development. is a design tool used by top companies like Microsoft and PayPal, and also by software houses and freelance designers [54].

2.3 Summary and conclusion

In the above reviews, software technologies and software tools were separate, unlike previous surveys which did not differentiate between them. This makes it much easier to select, develop, combine, or enhance a certain technology independent of the tools to implement it.

It is also clear from the literature survey that the state-of-the-art RWD technologies are directed toward the responsive layout modules CSS Flex and the CSS Grid [1]. That has become so prevalent in recent years. Both modules possess very powerful CSS page planning tools and are widely supported by platforms such as browsers and libraries. However, the literature did not include deep comparisons showing how and when to use each of them and left the choice up to the developer.

This thesis performs a thorough analysis of both technologies, identifies with practical verifications their differences and similarities, and determines when and how to use each of them. As mentioned in [24] appropriate technology may fail due to bad implementation, In its practical part, **this thesis** provides a guide model showing how each technology can be best implemented. It is also clear from the literature review and from the varieties of software tools that RWD is not just applying a specific technology [27], but rather it is a process of integrating a set of technologies using appropriate tools. CSS flexbox and grid technologies were combined in the famous Bootstrap version 5 [5]**.** But as mentioned in our problem statement (sec 1.5) their integration does not allow to correct or update the site without restarting and reloading a huge amount of unnecessary codes. The combined model presented in **this thesis** does not suffer from this drawback as it is possible to correct or update the site without restarting or reloading any codes. Therefore, saving implementation time and space.

3 Analysis of CSS Flexbox VS Grid Technologies

Web applications have become more complex over the years and complex solutions such as “floats” have become inappropriate. It is now necessary to implement advanced layouts using new RWD technologies. As seen in Chapter 2, two new technologies have evolved, the **CSS Flexbox** and **Grid** with many characteristics that make them so powerful [22, 23].

This Chapter provides a complete analysis of both of them and verifies their important characteristics with practical examples. More importantly, both are compared in terms of similarities and differences showing their pros and cons. Many programs are developed here to show and practically verify the characteristics of both technologies. To save space only the output of these programs are included without the program codes because our main purpose here is to show the performance of each technology not how to code it. However, the programs are available for anyone who might be interested to use or verify them.

3.1 CSS Flexible Box Layout

CSS Flexible Box Layout is a CSS module optimized for user interface design, and for the layout of items in one dimension, which makes it a great technology to use for responsive design systems [55]. The main idea behind the flex layout is to give the container the ability to alter its items’ width, height, and order so that it can best fill the available space to accommodate any screen size.

Flexbox elements are arranged according to the main axis. This axis can be horizontal or vertical, that is, the elements can be divided into rows and columns. The content can be stretched, reduced, turned the other way around, and rearranged on the pages. It can also be specified vertically and horizontally, one or more lines can line it up, or the content can be adjusted according to the space to be used. With Flexbox, the elements on the pages can be adapted to behave differently depending on the screen size, which makes it well-suited for implementing responsive pages [38].

It depends on the concept that there is a “flex container “and elements inside this container (the flex items). Flexbox items are the immediate children in a flex container. A flex container expands items to fill available free space or shrinks them to prevent overflow [65]. When elements are laid out as flex items, they are laid out along two axes [57] as shown in the following output of a general CSS Flex Model**:**

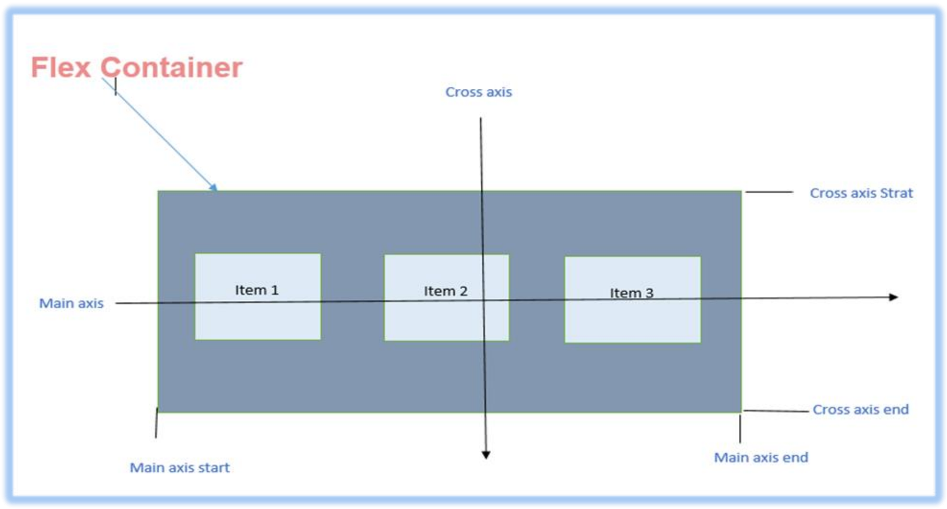


Figure 9: Shows the implementation of a program inside a flexbox container.

Moreover, Flexbox is a module involving a whole set of properties, some of them are related to the container **(“Flex container”)** whereas the others are related to the children **(Flex items)**. Properties have to be set up with appropriate values to be realized as shown in the following sections:-

3.1.1 Properties of Flexbox container

The following seven properties are related to the Flex container:

I. Display,
II. Flex-direction,
III. Flex-wrap,
IV. Flex-flow,
V. Justify-content,
VI. Align-content,
VII. Flexbox Gap.

I. Display:-

When using Flexbox, the parent element is called a flex container. For creating a flex container, the display property is used and a flex value is passed.

```
.container {
   display: flex;
 }
```

II. Flex-direction

The direction of how the child elements are shown inside the flex container can be changed using the flex-direction property. With flexbox, one can add the values of CSS properties to the element, which defines how the child elements should be positioned.

```
.container {
  Flex-direction: row | row-reverse | column | column-reverse;
}
```

This property is verified by our coding which produced the following Results according to different values:-

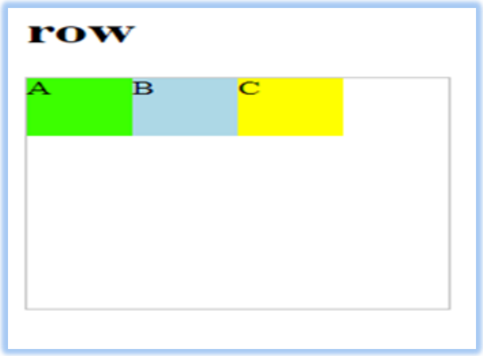
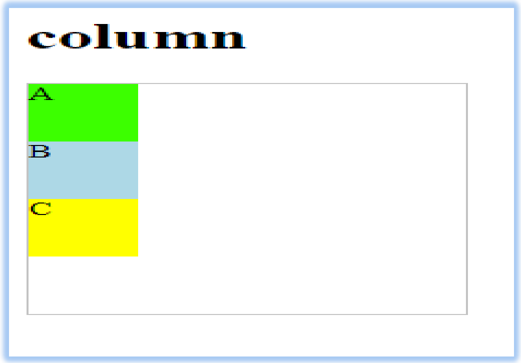
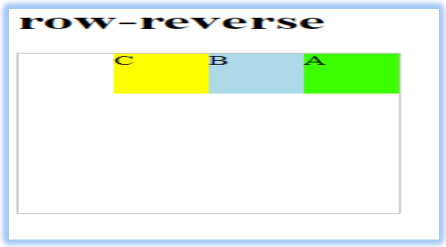
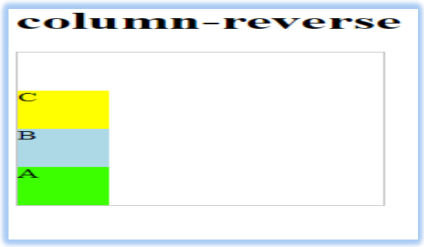
 **flex-direction: row**	 **flex-direction: column**
 **flex-direction: row-reverse;**	 **flex-direction: column-reverse;**

Figure 10: Program results that display values for Flex-direction: the property

III. Flex-wrap

The flex-wrap property specifies whether the flex items should wrap or not. The flex-wrap property is a sub-property of the Flexible Box Layout module, It defines whether the flex items are forced in a single line or can be flowed into multiple lines, Its function is to make the child elements move to the next line if there is not enough space to display the child elements in their normal size, Its direction depends on the direction of the main axis.

This property takes the following values:-

```
.container {  flex-wrap: nowrap | wrap | wrap-reverse;}
```

Example: Let us create a container that is 500 pixels wide with six elements placed in it, each element is 33 pixels wide. **Container width > Sum width Elements.**

```
#content {    width: 200px;    height: 200px;    display: flex;    flex-wrap: wrap ;  }    .box {    width: 33px;

   height: 50px;  }
```

Here the width of the sum of the elements is less than the container, no wrapping occurs, and the elements remain on the same line Total of **198 items < 200** container width.

After implementing an example, the results are as follows:

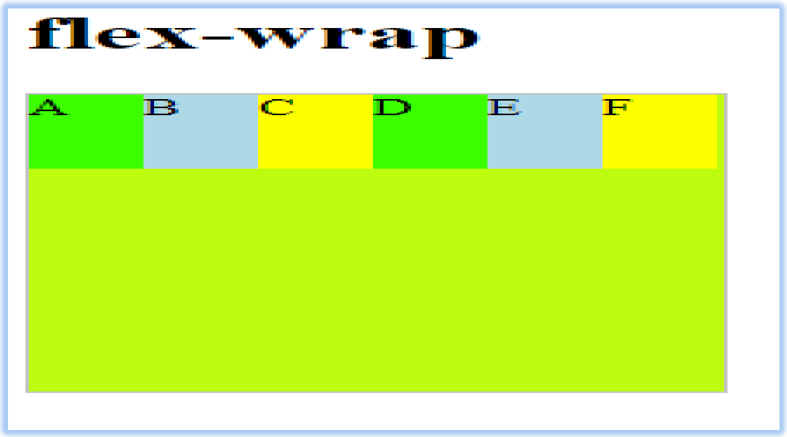


Figure 11: Program results showing the wrap property

Example: The same as the last example but the width of the container is 200 pixels, and the width of each element is 50 pixels, **Container width <Sum width Elements:**

```
     #content {    width: 200px; height: 200px;    display: flex;    flex-wrap: wrap ;  }    .box {    width:
0px;    height: 50px; }
```

After implementing this example, the results are as follows:

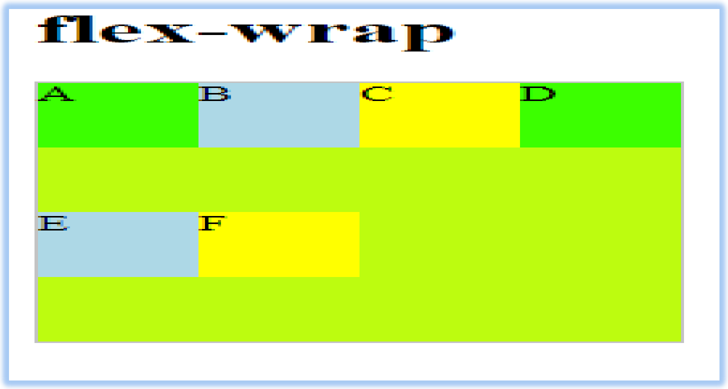


Figure 12: Program results showing the wrap2 property

Note the creation of a new line automatically, provided that the property flex-wrap is given**:** because the default value of this property is no wrap (Total 250 items > 200 container width).

IV. flex-flow

This property combines the two previous properties in the sense that it is an abbreviation:

- The first value is the flex-direction value.
- The second value is the flex-wrap value.

```
.container {
  flex-flow: column wrap;
}
```

V. justify-content

This property uses set content in Flexbox containers to change the alignment of flex items on the main axis (x-axis to start, y-axis if flex-direction: column) The flex-direction can be either aligned row- or column-wise. This property takes the following values:-

```
justify-content: flex-start | flex-end | center | space-between |  space-around | space-evenly
```

This property is verified by our coding which produced the following Results according to different values:-

- If the flex-direction: row elements will be aligned on the X-axis.

```
flex-direction: row
```

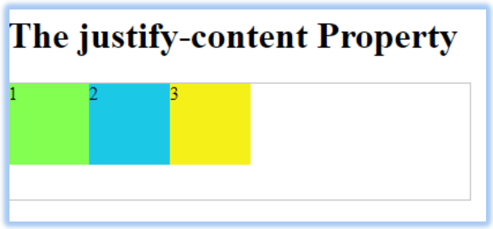
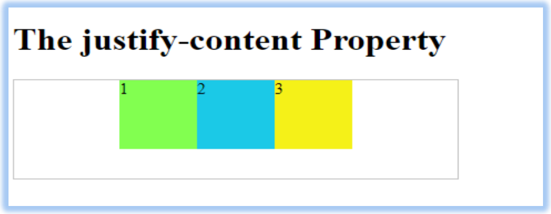
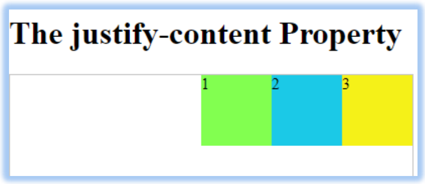
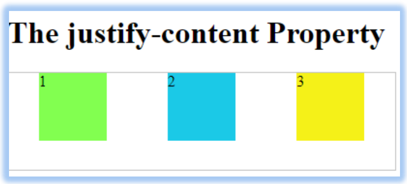
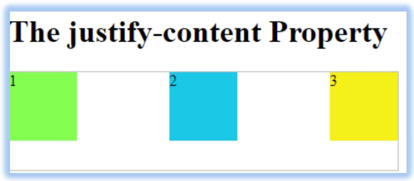
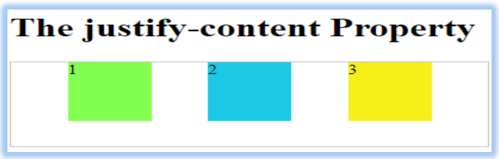
 **flex-start**	 **center**
 **flex-end**	 **space-between**
 **space-around**	 **space-evenly**

Figure 13: Program results that display values for justify-content the property

This property is verified by our coding which produced the following Results according to different values:-

- If the flex-direction: column; elements will be aligned on the Y-axis

- flex-direction: column ;

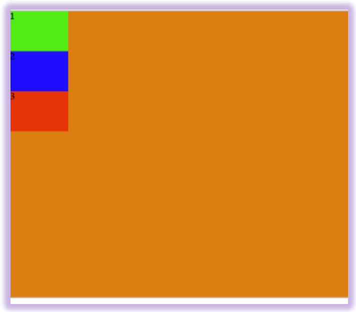
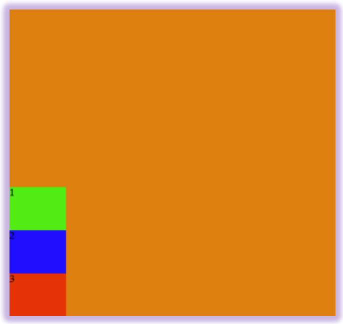
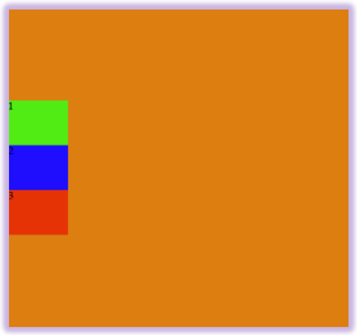
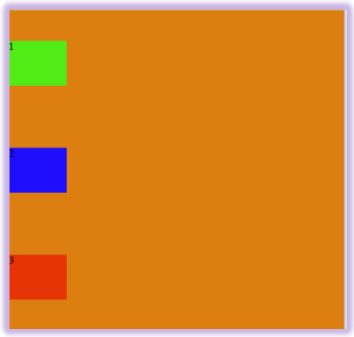
justify-content: flex-start; 	**justify-content: flex-end;** 
justify-content: center; 	**justify-content: space-around ;** 

Figure 14: Program results that display values for justify-content2 the property

VI. Align-content

Reverse justify-content in terms of axis because it is used to align the child elements of the secondary axis. This property takes the following values:-

```
.container {
  align-items: stretch | flex-start | flex-end | center | start | end |
}
```

- if flex-direction: row; The elements will be aligned on the Y axis.
- if flex-direction: column; The elements will be aligned on the x-axis.

Example: created a container of 3 items. The condition is that these items are placed in the middle of the container, that is, the center:-

```
flex-flow: row wrap ;

justify-content: center;

align-content: center;
```

The elements are placed in the middle of the container, and any elements are controlled horizontally and vertically through these properties **justify-content: center**; **align-content: center.**

After implementing an example, the results are as follows:

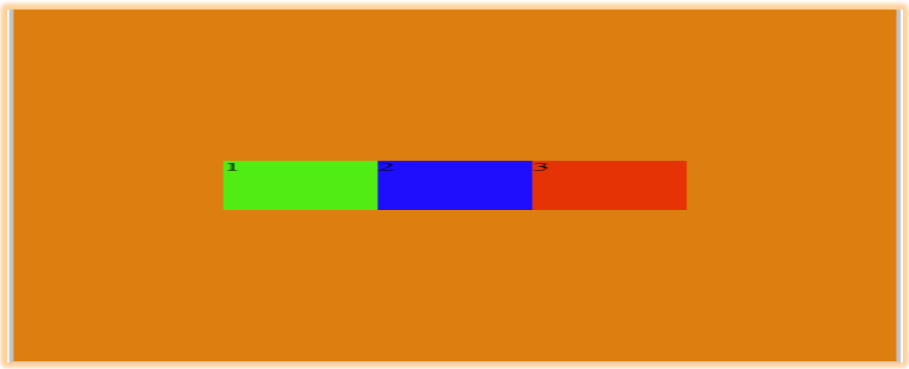


Figure 15: Alignment content example execution program

VII. Flexbox Gap

The gap property in CSS is a shorthand for row-gap and column-gap, specifying the size of gutters, which is the space between rows and columns within the grid, flex. In 2021 a new CSS trick is called the flexbox gap [58]. This property is borrowed from Grid CSS and is called grid-gap. The gap is used to create space between cells.

```
flex. Container{
gap | row-gap | column-gap
}
```

Example: Flexbox layout 50px gap between rows, 60px gap between columns: After implementing an example, the results are as follows:

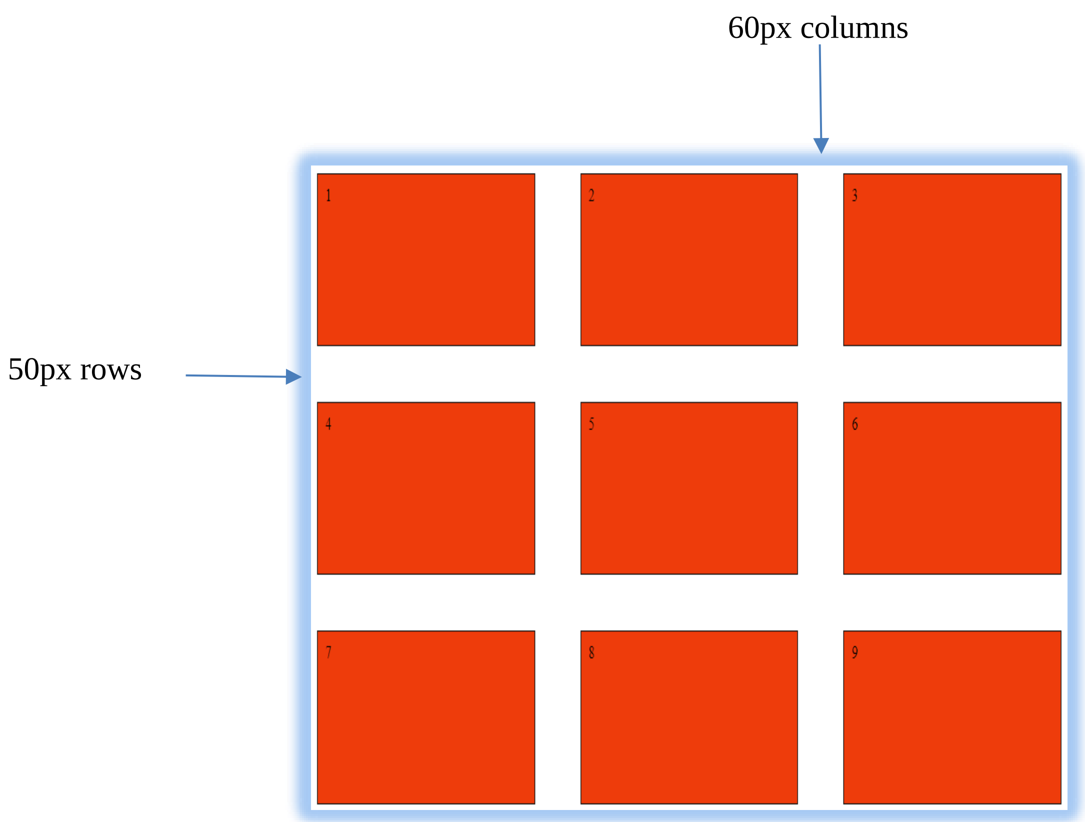


Figure 16: An implementation of a program that shows the Flexbox Gap

3.1.2 Properties of Flex items

Flexbox items are the immediate children in a flex container. They enjoy the following properties that must be given the desired values:

I. Flex-grow,
II. Flex-shrink,
III. Flex-basis,
IV. Flex: “0, 1, auto “,
V. Order.

I. Flex-grow

The flex-grow CSS property sets the flex grow factor of a flex item's main size. This property is used when the width of the container is greater than the sum of the width of the child elements . This property takes values as either 0 or 1.

Example: Create a container with 4 elements so that the width of the container is greater than the sum of the width of the elements, the width of the container is put at 1000 px, and the width of each element at 100 pixels.

CSS

```
.flex-container {  display: flex;}.flex-container > div {  width: 100px;}.item{    flex-grow: 0;}
```

The default value of flex-grow is 0, which means that the child element is not expanding.

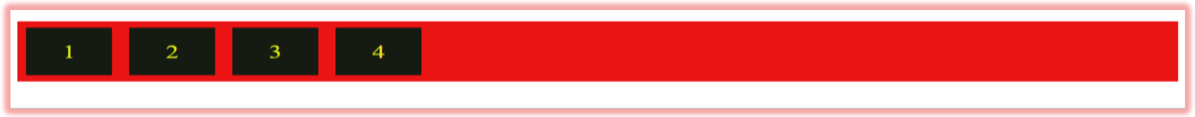


Figure 17: Program results that display values for flex-grow the property

Thus, there is a space inside the container, and if the child elements are to be expanded (grow) so that there is no space inside the container, the flex-Grow property is used.

When elastic growth is: 1; This means this element is allowed to expand to take up the remaining space of the container.

CSS

```
item{    flex-grow: 1;}
```

Result

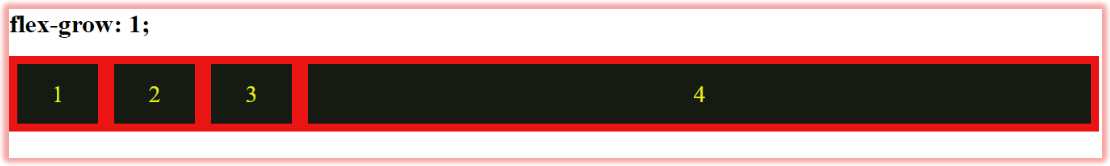


Figure 18: Program results that display values for flex-grow2 the property

II. Flex-shrink

The flex-shrink CSS property sets the flex shrink factor of a flex item. This property is used when the sum of the width of the child elements is greater than the width of the container.

This property takes values as either 0 or 1 to be used when the child element is to be shrined, because the child elements are large, they shrink automatically.

Example: Create a container with 3 elements, so that the width of the container is less than the sum of the width of the elements, the width of the container is put as 300 pixels and the width of each element as 120 pixels:

CSS

```
.flex-container {  display: flex;width: 300px;flex-wrap: nowrap; }.flex-container  div {    width: 120px;

 margin: 10px;  text-align: center;  }
```

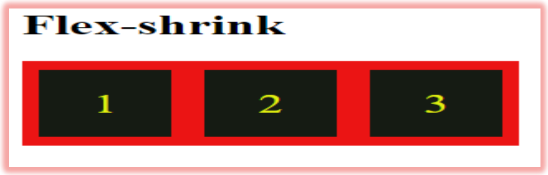


Figure 19: Program results that display values for flex-shrink the property

Note that the width of the element cannot be known, and it shrinks automatically by the flexbox. To keep the width of the previously defined element, the following code must be added:

```
.flex-container {    flex-shrink: 0;}
```

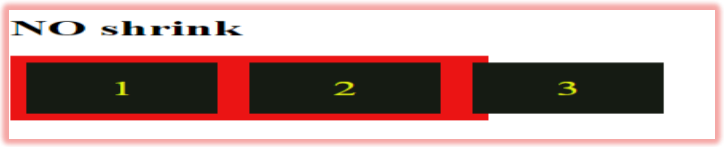


Figure 20: Program results that display values for flex-shrink2 the property

III. Flex-basis

The flex-basis property determines the width or height of the child element, depending on the flex-direction.

- If the **flex-direction: row;** the flex-basis determines the width of the element.
- If the **flex-direction: column;** the flex-basis determines the height of the element.

The default value is **flex-basis: auto**; what this means is that the element is in its natural width or length.

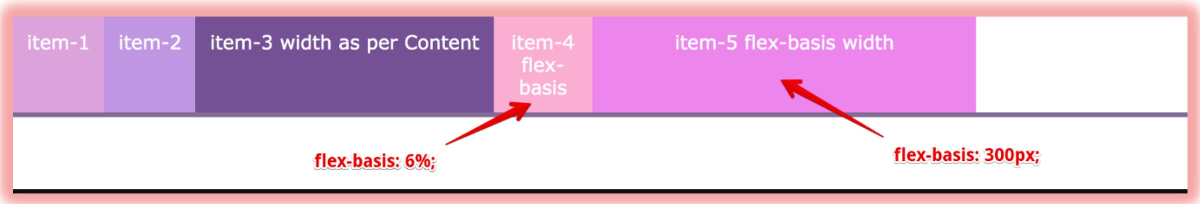


Figure 21: display values flex-direction of the property

IV. Flex: “0, 1, auto”

This property is an abbreviation for specifying the flex-grow flex-shrink and flex-basis values instead of writing each separate code. The default values are:- Flex: 0, 1, auto :

0: This means that the element is not allowed to extend (grow) if there is a space inside the container.

1: I.e. the element is allowed to shrink (shrink) if the width of the element is greater than the width of the container. Auto: This means that the element is in normal width or length.

V. Order

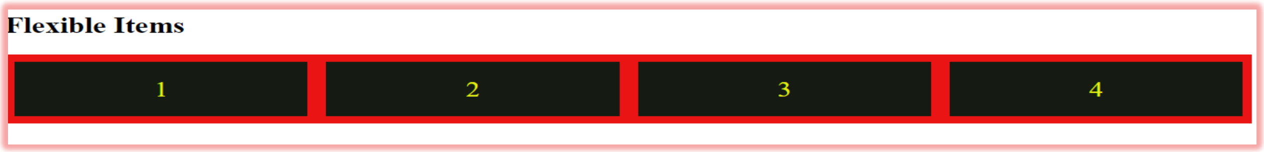
This property is used to arrange the elements, and the element that contains the least value is at the beginning of the order, All elements have a default value of 0.

Example: Here create a container with 4 elements, and what is required is to change the centers of the elements, for example, the fourth element replaces the first element: -

CSS

```
.flex-container {  display: flex; }.flex-container > div {   width: 500px;  margin: 10px;  text-align: center;

 line-height: 75px;  font-size: 30px;}
```



Flexible Items

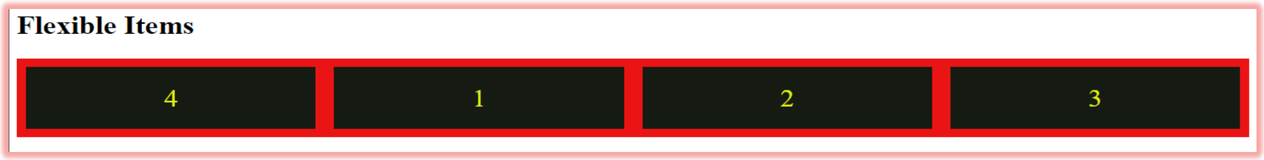
1 2 3 4

Figure 22: Program results that display values for the order the property

The default value of all elements is zero. Here, to move element number 4 to the beginning of the container, the following cod is added:

```
.item{  order: -1;}
```

The element with a lower value is the element that is at the beginning of the order, the fourth element takes the lowest value:-



Flexible Items

4 1 2 3

Figure 23: Program results that display values for the order2 the property

3.2 CSS Grid Layout

The CSS Grid Layout Module, or CSS Grid, is a two-dimensional layout system, which can work with rows and columns together [59], i.e. horizontal and vertical lines as shown in Grid Model (Fig 24) below. This provides a lot of possibilities to build more complex and organized designs and makes it much easier to position items in the container in any area without having to mess with the HTML markup [60].

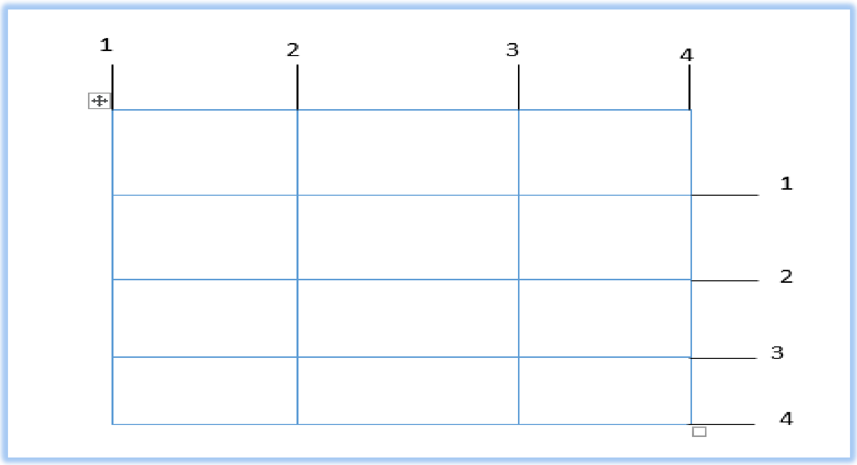


Figure 24: shows the space divided into rows and imaginary columns (invisible).

Like Flexbox technology, it is a container with elements inside as in figure 25

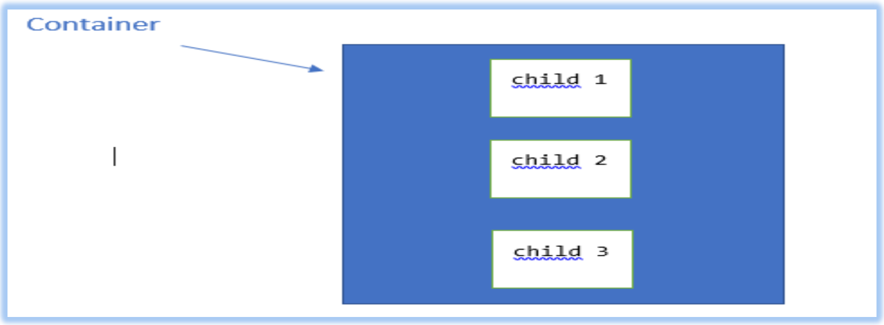


Figure 25: Display the implementation of a program inside a Grid container.

This layout grid model helps in placing elements accurately as well as rearranging them according to different screen sizes, which makes it effective for responsive design.

Grid layout divides the size of the paths as fixed or flexible: Fixed sizing can be created using **px** for example. Flexible sizing can be created using a percentage **%**, i.e. a Fraction Unit **(Fr)** of the available space in the grid container [61].

Example: The distance between the columns must be equal in size and fill the available space. In static partitioning, each column is set to 100/4 = 25% wide. Fig 23 shows a page divided into four columns.

```
grid-template-columns: 25% 25% 25% 25%;
```

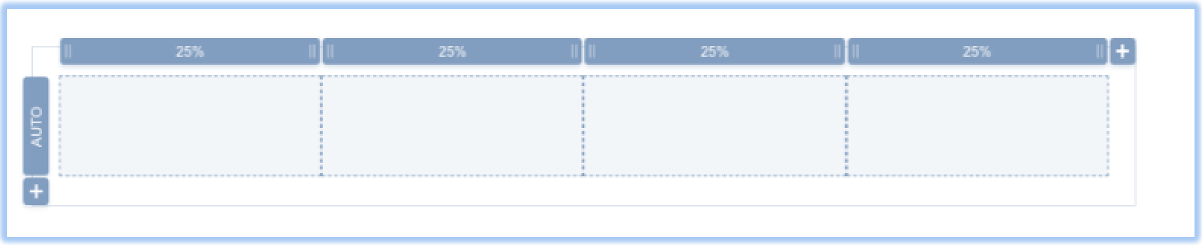


Figure 23: Displays page division

and the following code gives the same result, with split Fr provided by the grid technology which makes it simple and responsive:

```
grid-template-columns: 1fr 1fr 1fr 1fr;
```

CSS Grid technology, like CSS Flex technology, is a complete module with a full set of properties. Some are supposed to be set on the container (the parent item, known as "Grid Container") while others are supposed to be set on the children (known as "Grid Items").

3.2.1 Properties Grid Container

The following six properties are related to the Grid container:

- **I.** display.
- **II.** grid-template-columns.
- **III.** grid-template-rows.
- **IV.** Gap.
- **V.** justify-content.
- **VI.** align-content.

I. Display

To take advantage of the features of the grid, one can put the **(display: grid)** ; when using this property, one can control the elements vertically and horizontally at the same time; unlike the flexbox, in which one can only control the elements, either vertically only or only horizontally for the parent element:-

```
.container {
 display: grid ;
 }
```

II. The grid-template-columns Property

The grid-template-columns property defines the number of columns in a grid layout, and it can define the width of each column. The value is a space-separated-list, where each value defines the width of the respective column

Example: Here in the following code, specify the number of three dummy columns so that the width of the first is 100 pixels, the width of the second is 30% of the width of the parent element, and the third column occupies the rest of the space):-

```
Grid-template-columns: 100px 30% 1fr;
```

Shown in Figure 26:-

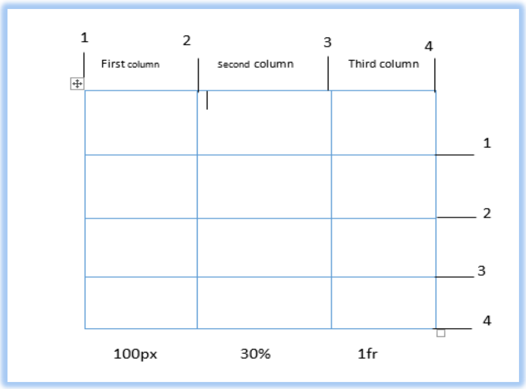


Figure 26: Displays the invisible partition method of the property grid-template-columns

Then split the page with all the values used in this technique (px, %, fr)The grid-template-rows Property

III. The grid-template-rows property

The grid-template-rows property specifies the number (and the heights) of the rows in a grid layout.

Example: specifies the height of each row Here in the following code, specify the number of three imaginary rows so that the height of each row is (the height of the first row is 100 pixels, the height of the second row is 110 pixels, and the height of the third row is 120 pixels):-

```
grid-template-rows: 100px 110px 120px;
```

Shown in figure 27:-

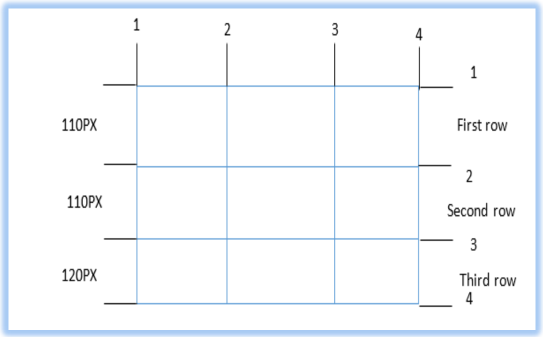


Figure 27:Displays the invisible partition method of the property grid-template-row

IV. The grid-gap Property

The grid-gap property defines the size of the gap between the rows and columns in a grid layout.

Example: Here a container with 6 elements is created and the page is divided into 3 columns through the gap property creating a space between the rows 50px and the rows 50px.

```
.grid-container {

   grid-gap: 50px 50px;  }
```

Which can be seen in the following implementation:

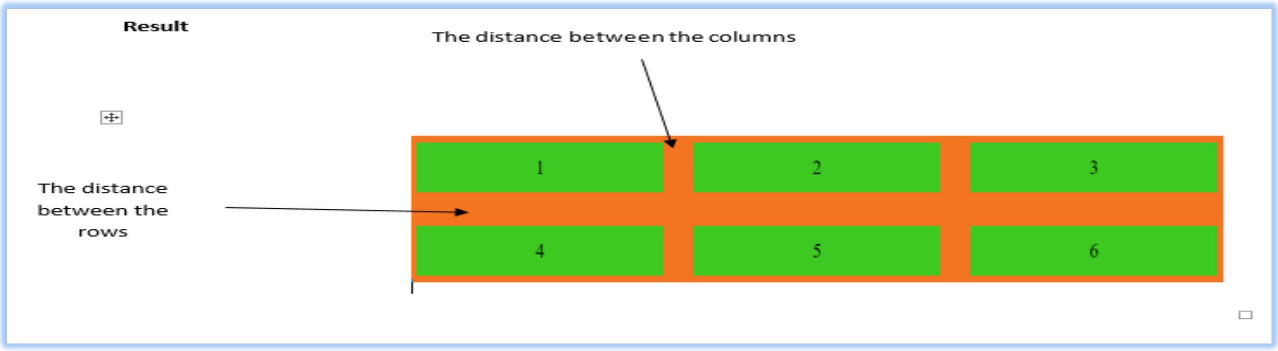

Result
The distance between the columns
The distance between the rows
1
2
3
4
5
6

Figure 28:An implementation of a program showing the gap property

V. The justify-content Property

Control the horizontal movement (on the x-axis) of all child elements these are the most important values of justify-content, This property takes the following values:

```
.container {
  Justify-items: start | end | center | stretch;
}
```

This property is verified by our coding which produced the following Results according to different values:-

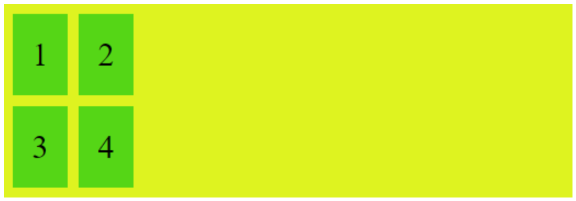
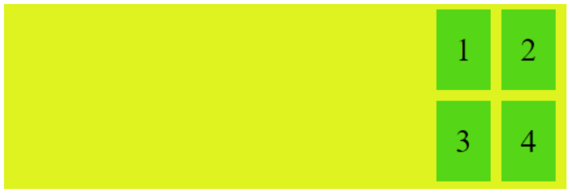
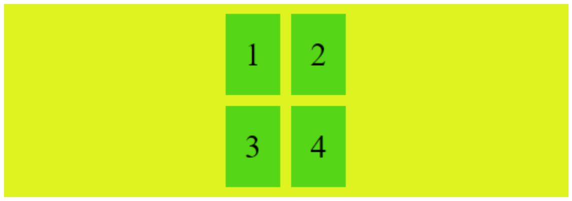
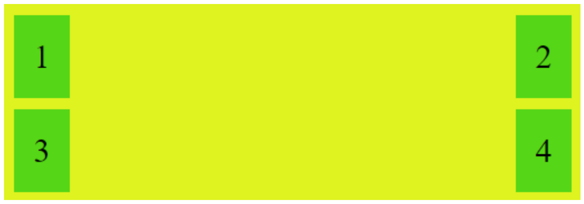
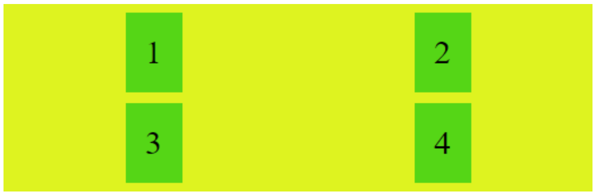
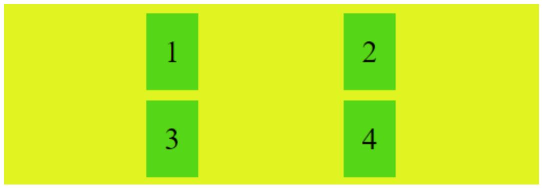
 **justify-content: Strat**	 **justify-content: End**
 **justify-content: Center**	 **justify-content: space-between**
 **justify-content: space-around**	 **justify-content: space-evenly**

Figure 29: Program results that display values to justify the content of the property

VI. The align-content Property

The align-content property is used to vertically align the whole grid inside the container.

```
.container {
  align-items: start | end | center | stretch;
}
```

This property is coded and verified with different values as follows:-

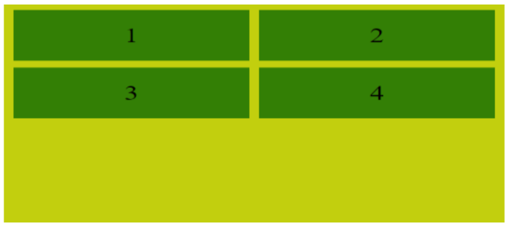
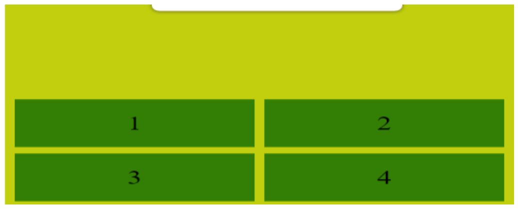
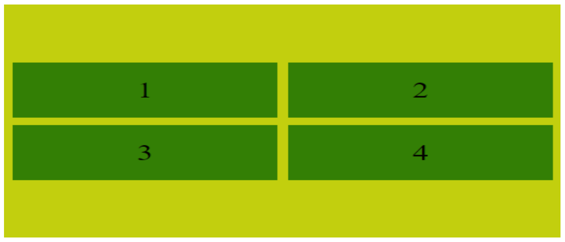
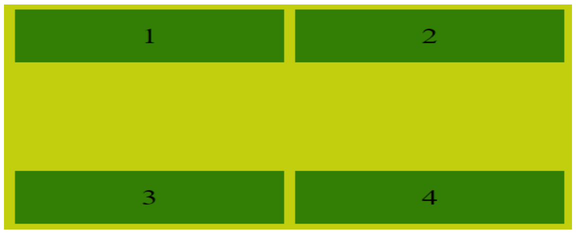
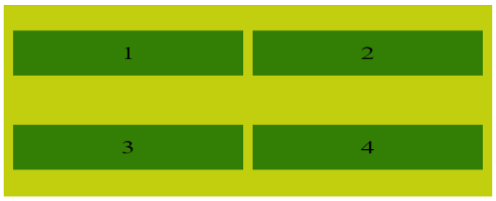
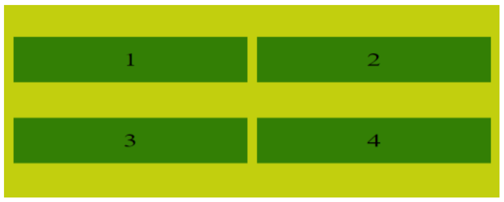
 **align-content :Strat**	 **align-content :End**
 **align-content :Center**	 **align-content:space-between**
 **align-content:space-around**	 **align-content:space-evenly**

Figure 30: Program results that display values for align-content the property

3.2.2 Grid Items Properties

Grid items are the immediate children in a flex container. They enjoy the following properties that must be given the desired values:

I. Grid-column.
II. Grid-row.
III. Grid Area.

I. Grid-column

Shorthand for start displaying item/ end of item display. This property takes the following values:

```
Grid-column: <start-line> / <end-line> | <start-line> / span <value>;
```

II. Grid-row

Shorthand for start Height item/ end of item Height. This property takes the following values:

```
Grid-row: <start-line> / <end-line> | <start-line> / span <value>;
```

Example: This interactive shows the two properties Grid-column: start displaying item/ end of item display; and Grid-row: start Height item/ end of item Height;

Let us create a container with 8 items and the with following conditions:

- Make "item1" start on column 1 and end before column 2.
- And Make "item1" start on column 1 and end before column 3.
- And Make "item1" start on row 1 and end before row 2.
- And Make "item1" start on row 1 and end before row 3.

A container has been created containing 8 elements and here the first element is allowed to be expanded using the property: Grid Column Property and Grid Description as shown in the

After implementing an example, the results are as follows:

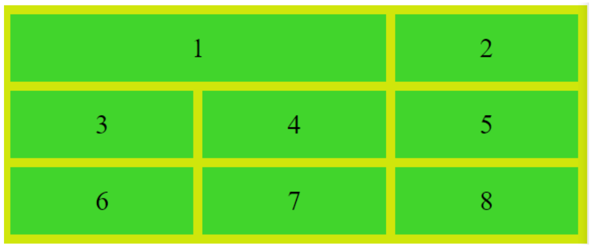
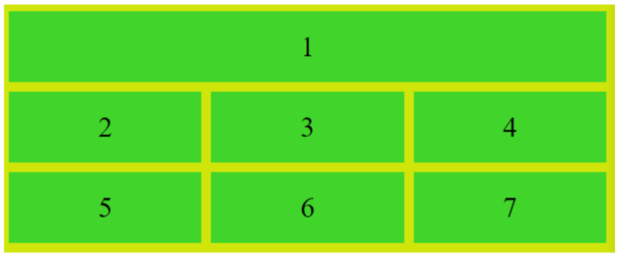
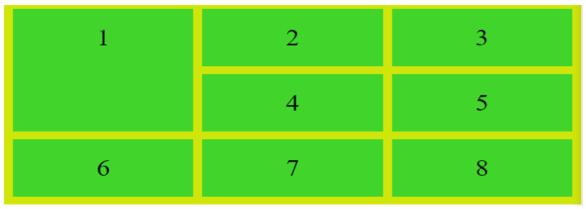
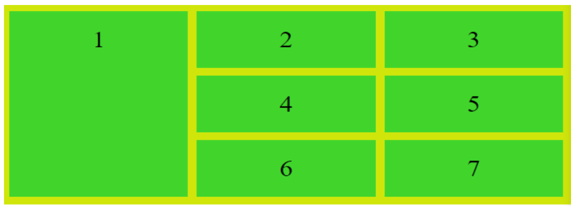
 **grid-column: 1 / 2;**	 **grid-column: 1 /3;**
 **grid-row: 1 / 2;**	 **grid-row: 1 / 3;**

Figure 31: It is an implementation of a program that shows Grid Items Properties

III. Grid Area property

The grid-area CSS property is a shorthand It is another way to use the grid technology. In this method the page is not divided by imaginary lines but by giving each element its name [62]. This property takes the following values:

- **grid-template-areas**

It is a combination of the two previous characteristics, that is, it takes rows and columns:

Grid-template-columns: auto; + grid-template-rows: auto;

- **grid-area**

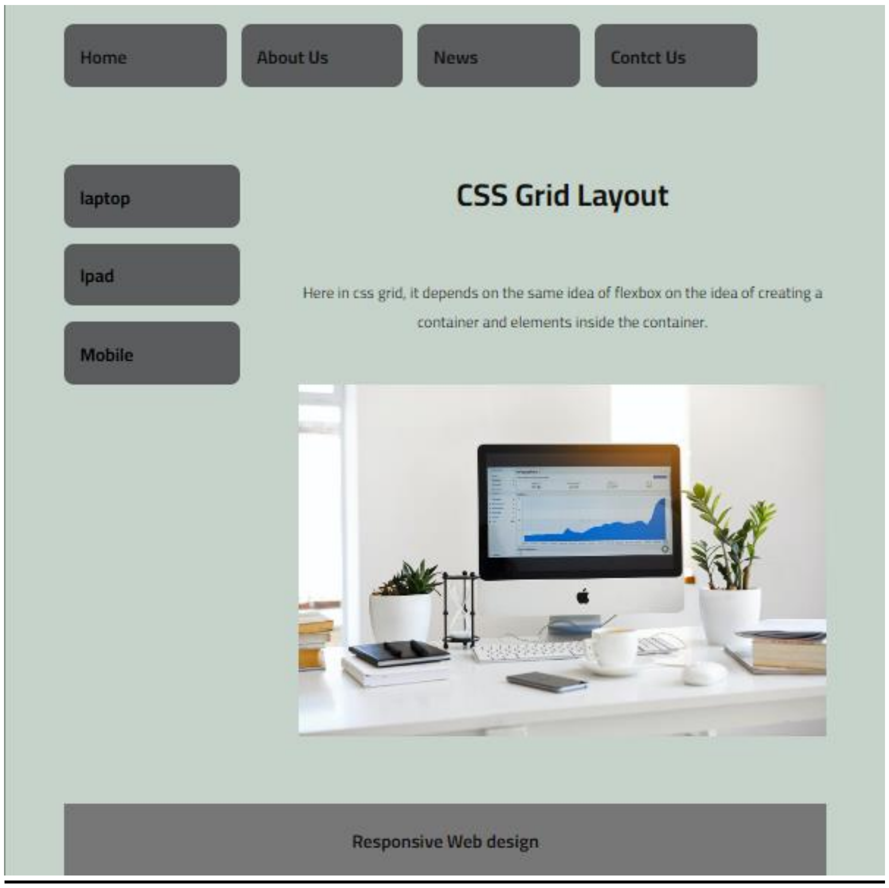
It is a combination of the two previous characteristics, that is, it takes rows and columns:

Grid-column: start displaying item/ end of item display; + Grid-row: start height item/ end of height item;

Example: In the below, the grid container contains two grid elements, named with the grid area property and then placed on the grid using the grid template areas.

After implementing an example, the results are as follows:

:



Here in css grid, it depends on the same idea of flexbox on the idea of creating a container and elements inside the container.

Figure 32: An implementation of a program shows the Grid area

3.3 Similarities of Flex and Grid?

CSS Grid and Flexbox are layout models that share similarities starting with the fact that they are responsive models; it is used for planning and is much more powerful than any planning technique that came before. They can expand and contract, they can center things, they can rearrange things, and they can align things [63].

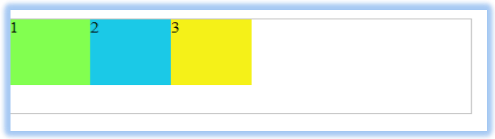
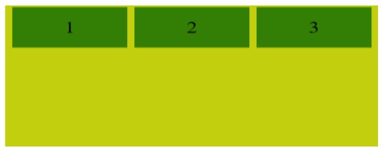
The following table shows the most important similarities between them, according to the examples previously studied: -

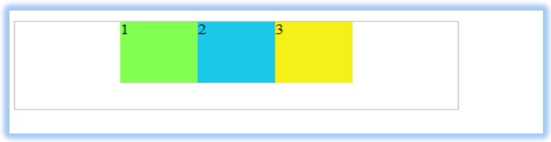
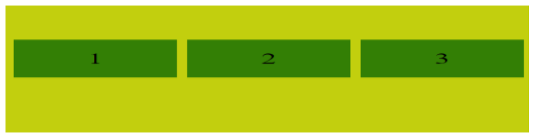
Similarities Summary

properties	CSS Grid	CSS Flexbox
the structure	Depends on the idea of the container and the elements inside the container	Also Depends on the idea of the container and the elements inside the container
justify-content	It defines the alignment along the main axis.	Also, It defines the alignment along the main axis.
align-content	It defines the alignment along the cross-axis.	Also defines the alignment along the cross-axis.
Page Responsiveness	Responsive Planning Model	Responsive Planning Model

Table 1: shows similitudes entre CSS Grid and CSS Flexbox:

Here are some values for similar properties of the two technologies, which are illustrated in the following figures:-

flex: justify-content: flex-start;
grids: justify-content: start

	flex: align-content: flex-start;
	grids: align-content: start;
	

Flex: justify-content: center;
Grids: justify-content: center;

	Flex: align-content: center;
	Grids: align-content: center
	

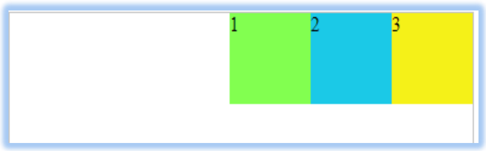
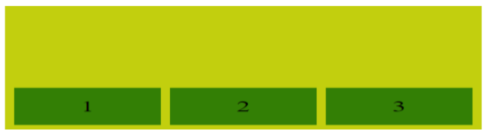
Flex: justify-content: flex-end;
Grids: justify-content: end;

	Flex: align-content: flex-end;
	Grids: align-content: end;
	

Figure 33: A program that shows similarities between CSS Grid and CSS Flexbox

3.4 Differences between Flex and Grid?

I. One-dimensional (1D) versus two-dimensional layout (2D).
II. Flexbox Wraps vs Grid Wraps
III. Content-First vs Layout-First
IV. Grid “Gap” Property vs Flexbox “Gap” Property

I. One-dimensional (1D) versus two-dimensional layout (2D).

The main difference is that one can use CSS Grid to create two-dimensional layouts. In contrast, can only use Flexbox to create one-dimensional layouts. That means can place components along the X- and Y-axis in CSS Grid and only one axis in Flexbox. [63].

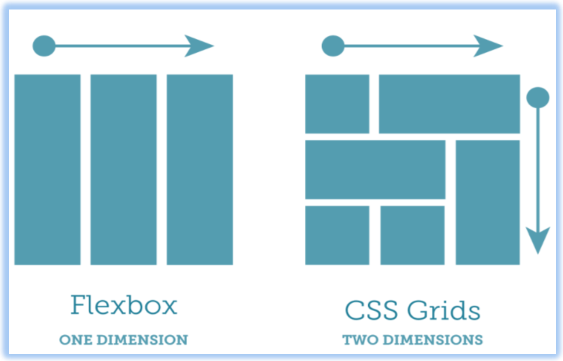


Figure 34: It shows the difference between CSS Grid vs Flexbox.

- **In Flexbox**

Flexbox is great for making layouts in one dimension, when the goal is to arrange elements in either row or column flex-box makes it simple and does an amazing job. It will give you more flexibility than CSS Grid. It’ll also be easier to maintain and require less code.

For example:-

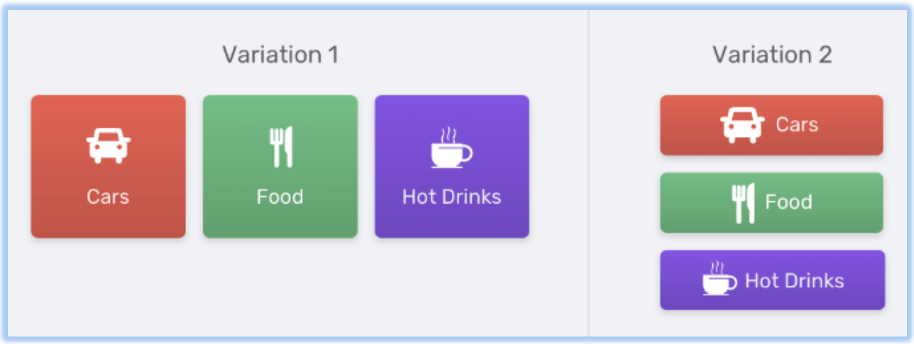


Figure 35: shows Flexbox's one-dimensional layout.

- **In CSS Grid**

When laying out is needed to be in two dimensions and arrange items in well-defined rows and columns then the Grid is the best. In this case, CSS Grid will give you more flexibility, make your markup simpler and the code will be easier to maintain.

For Example :-

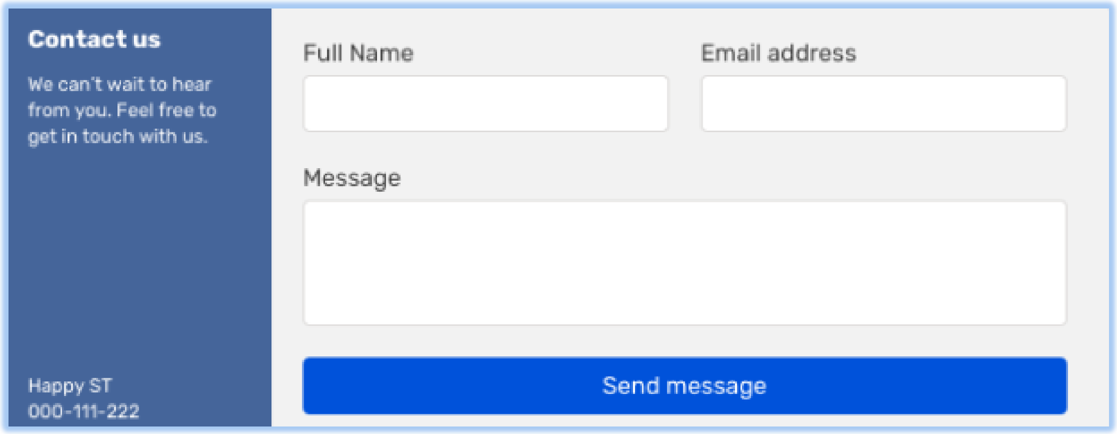


Figure 36: shows the Grid two-dimensional layout.

II. Flexbox Wraps vs Grid Wraps

Flexbox can also arrange elements in 2D by using the optional wrap property. But that **wraps** the content to the next line when it overflows but it won't necessarily be a well-defined column or rows. The content can expand and take its own space depending on how many elements are along with it in that row/column. Grid always arranges elements in well-defined rows and columns [64].

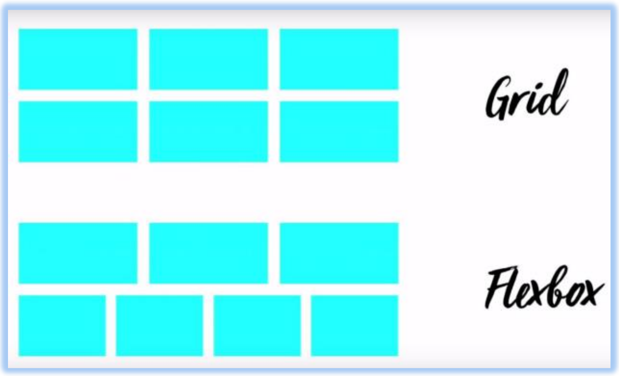


Figure 37:shows the difference between the two techniques in terms of wrap

When the total width of items inside the container is greater than the width of the container, in that case, both the layout models have the option to wrap the items in a new row. Although both models have ways of wrapping, each model handles wrapping very differently.

Example: distribute five elements over two columns, first with Flexbox and then with CSS Grid.

- **In Flexbox**

The container has been given flex property values so that it can grow and shrink, e.g. the width of each element 150 px and the width of the container 600 px so that the sum of the elements is greater than the width of the container to fulfill the wrap condition.

will also set the flex-wrap property to wrap, so that if the space in the container becomes too narrow to maintain the flex base, the elements will wrap in a new row.

Because of the flexible behavior of Flexbox, the first three span the first row, and the other two go into the next line. Both of them take up as much space as possible Thus, they do not align with the elements in the first row.

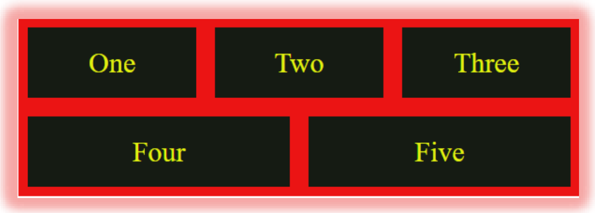


Figure 38: A program that displays wrapping elements using flexbox technology

So, a common question is how to make these elements align each element on top and one on the bottom. That is, to control the alignment by row and column, the Grid must come in.

- **In CSS Grid**

define three columns, and the elements use cells aligning to these columns. That means that they align perfectly.

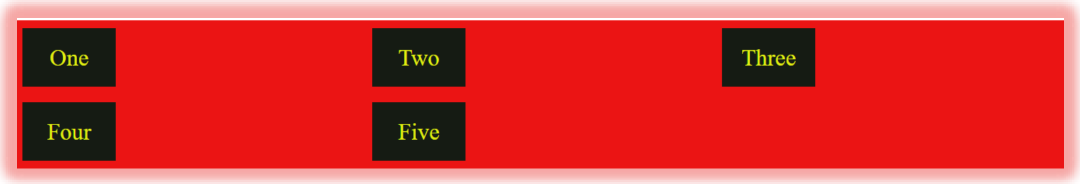


Figure 39: A program that displays wrapping elements using Grid technology

The most important thing to remember is that items in Flexbox lose the context of the previous row or column when they are pushed into a new row or column. That is because Flexbox is one-dimensional.

In CSS Grid, however, items don't lose their context when they're pushed down, because they're still part of the grid you've defined before. Items will fall along the grid lines.

III. Content-First vs Layout-First

Another major difference between Flexbox and Grids is that the Flexbox works on content while the Grid is based on the layout [65].

- **In flexbox**

The layout, the size of a cell (flex-item) is defined inside the flex-item itself Flexbox layout is calculated after its content is loaded.

CSS

```
.parent {
   display: flex;
  flex-flow: row wrap;
}
.child {
  flex: 1 1 auto;
  }
```

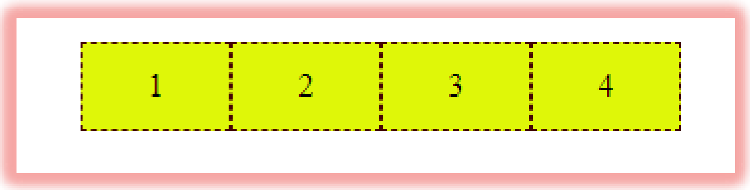


Figure 40:Shows the order of the items using Flexbox technology

- **In CSS Grid**

The size of a cell (grid-item) is defined inside the grid container, CSS Grid requires to define your layout first. It's container-based, and the layout is calculated regardless of the content put inside the containers**.**

CSS

```
.parent {
  display: grid;
 grid-template-columns: 1fr 1fr 1fr 1fr
. }
```

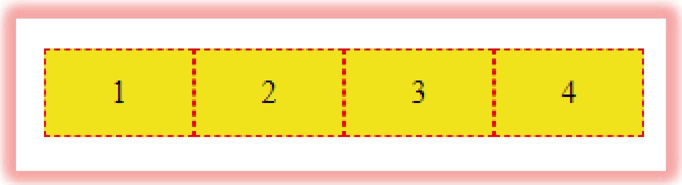


Figure 41: Shows the order of the items using grid technology

IV. Grid “Gap” Property vs Flexbox “Gap” Property

There was no flexbox Gap in the past, only in the grid. But after updates in recent years, this feature appeared to exist even in Flexbox [66] and will display the differences between them**.**

- **In Flexbox**

The Gap feature in Flexbox layouts is more complicated.

Example:-

CSS

```
 .row {      display: flex   }
```

By default, this gives us a row flex container, which means items within the container are stacked from left to right on the same horizontal line.

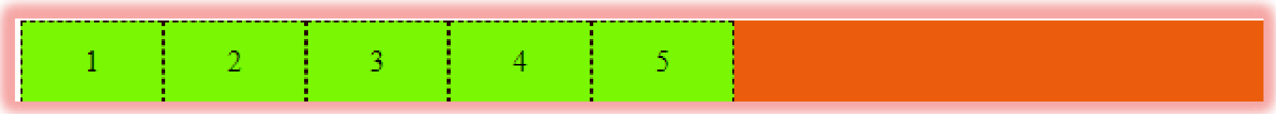


Figure 42: Shows the order of the items using Flexbox technology

In this case, the column gap is applied between items and the row gap does nothing. That is because there is only one line (or row). Now add the gap between items:

```
.container {    display: flex;    column-gap: 10px;  }
```

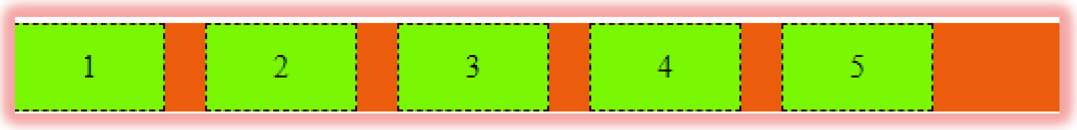


Figure 43: Shows the gap feature to use flexbox

Now the flex orientation of our container must change to a Column, which stacks the elements vertically, from top to bottom, using the following:

CSS

```
.container {    display: flex;    flex-direction: column;    column-gap: 10px ;}
```

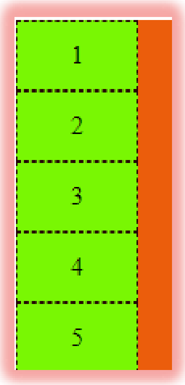


Figure 44: Shows the gap feature to use flexbox2

The gap disappeared. Even if the column gap did add space between items when the container was in a row direction, it does not work anymore in the column direction.

A row gap is used to get it back, or the gap shorthand with one value could be used, which would apply the same gap in both directions and, therefore, work in both cases.

CSS

```
.container {  display: flex;  flex-direction: column;  gap: 10px;}
```

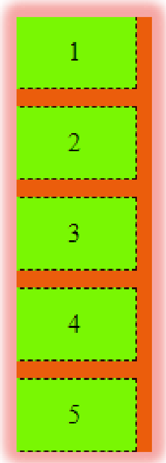


Figure 45: Shows the gap feature to use flexbox3

So, to summarize, Column-gap always works vertically (assuming the default writing mode), and row-gap always works horizontally. This does not depend on the direction of the flex container.

Another example where line wrapping is involved:

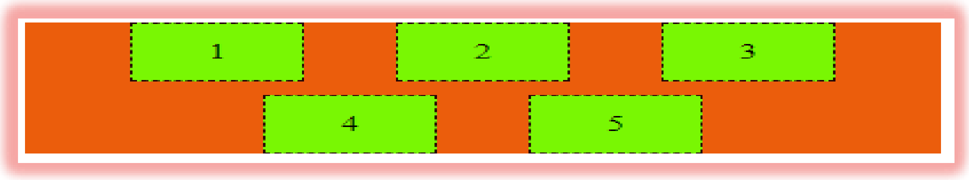


Figure 46: Shows the gap feature to use flexbox4

In this case, the column gap is still applied vertically between items, and the row gap is applied horizontally between the two flex lines.

There's one interesting difference between this and the grid. The column gaps do not necessarily align across flex lines because of the property ” justify-content: center” which centers items within their flex lines. So, it can be seen that each flex line is a separate layout where gaps apply independently of other lines.

- **In Grid gaps**

It is simple here because, by default, columns are vertical, and rows are horizontal, just like in a table. So it’s easy to remember where column-gap and row-gap apply.

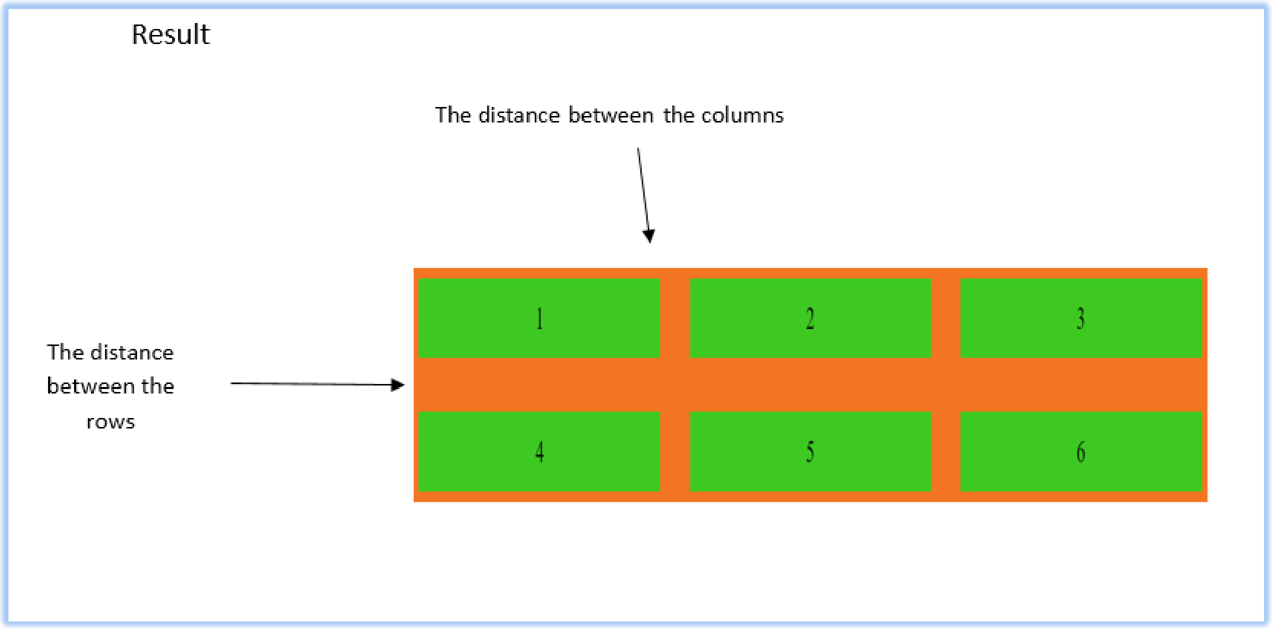

Result
The distance between the columns
The distance between the rows
1
2
3
4
5
6

Figure 47: Shows the gap feature to use a grid

Comparison summary

Element	Flexbox	Grid
Dimension	Flexbox is designed for one-dimensional layouts X or y	Grid is designed for 2D layouts. Rows and columns. X and y
Common	Flexbox has become very popular among front-end developers in the last couple of years.	Grid is much newer than Flexbox and has slightly less browser support.
Planning	With Flexbox, it's hard to predict behavior at certain viewports and you can get surprising results.	Grid always arranges elements in well-defined rows and columns.
Flexibility	Flexbox is surprisingly simple and works great for all browsers; It will give you more flexibility than CSS Grid. It will also be easier to maintain and will require less code	It is flexible but planning can anticipate what happens on all screen sizes.
Approach	Flexbox is content-based and it listens to the content and adjusts to it.	Grid operates more on the layout level and it is container-based.

Table 2 shows CSS Flexbox vs. CSS Grid: Comparison Table

3.5 CSS Grid. VS. CSS Flexbox layout systems

As indicated in our literature review **(Chapter 2)** implementation methods can make the technology succeed or fail [24]. This chapter provides a clear implementation plan as a complete guide model to ensure the successful implementation of The CSS Grid and CSS Flexbox. Each implementation model spells out the advantages and disadvantages of the used technology, which helps the developer, decide when to use and not to use it.

3.5.1 Creation of all elements:-

This step is common to both CSS Grid and Flex technologies. The default layout of most sites on the internet consists of a header, which is the" top header", navigation menus, centered" at the top or on the sides menu", and various sections in the middle. , And the sidebar side menu and footer" at the bottom of the footer page.

HTML

```
<header></header>
<div >
   <section class="main"></section>
   <aside class="sidebar"></aside>
</div>
<footer></footer>
```

When composing elements in HTML, the elements are vertical, as in the following figure -:

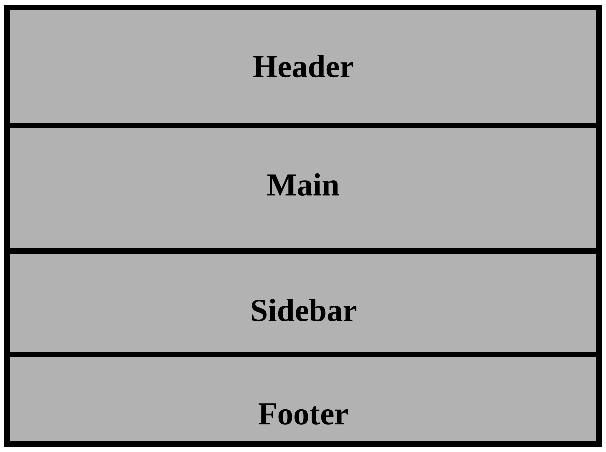


Figure46:*Shows the default shape of the elements*

3.5.2 Website design for large screens (computer): -

- **Design using Flexbox Technology**

Step 1: -

The following code allows using the properties of the flex technique where the default position of the elements is a row and can be changed to a column using the direction property.

CSS

```
.container {  display: flex;  flex-direction: column;}
```

Step 2: -

Make the main section and sidebar stand next to each other and you cum in an inner container.

HTML.

```
<header></header>
<div class=" main-and-sidebar">
   <section class="main"></section>
   <aside class="sidebar"></aside>
</div>
<footer></footer>
```

Step 3:-

To take advantage of the properties of the Flex the following code is written in the CSS file and the elements are automatically in row mode**.**

CSS

```
.main-and-sidebar-wrapper {    display: flex;    flex-direction: row;}
```

Step 4: -

Make the elements in the main list a row instead of a column and make an equal distance between the elements.

CSS

```
.flex-container {    display: flex;    justify-content: space-between;  }
```

Step 5:-

One makes the items in the side menu a column instead of a row.

CSS

```
   flex-direction: column;
```

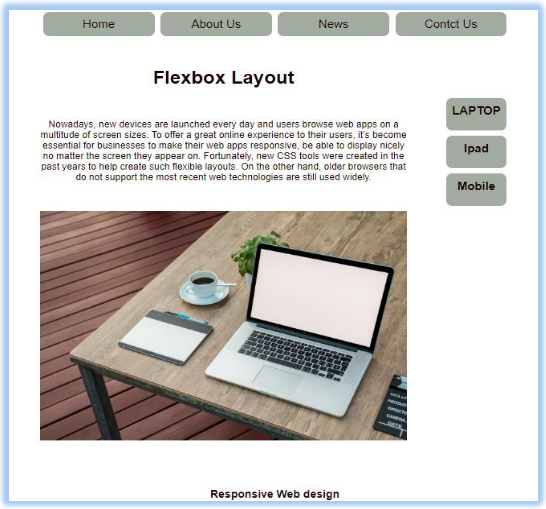
Step 6:-

The size of the main section and the sidebar is set. So that the main content is three times larger than the size of the sidebar.

CSS

```
.main {    flex: 3;    margin-right: 60px;}.sidebar {   flex: 1;}
```

The result is as follows:-



Flexbox Layout

Nowadays, new devices are launched every day and users browse web apps on a multitude of screen sizes. To offer a great online experience to their users, it's become essential for businesses to make their web apps responsive, be able to display nicely no matter the screen they appear on. Fortunately, new CSS tools were created in the past years to help create such flexible layouts. On the other hand, older browsers that do not support the most recent web technologies are still used widely.

Responsive Web design

Figure 48: shows flexbox layout for Desktop

- **Design Using Grid Technology**

Step 1:-

It is better to plan using the grid zones property due to the ease and clarity that this technology provides in planning, in this way, cells are created in the grid and assigned their names as follows.

CSS

```
header {
   grid-area: header;
}
.main {
   grid-area: main;
}
.sidebar {
   grid-area: sidebar;
}
footer {
   grid-area: footer;
}
```

Step 2:-

This may define the areas on the grid almost as if drawing them.

CSS

```
.container{
  display: grid;
  grid-template-columns: 1fr 3fr;
  grid-template-areas:
      "header header"
      " sidebar main"
      "footer footer";
  grid-gap: 60px;
}
```

The page was divided into two columns and three rows, the width of the second column is three times the width of the first column, and the areas are reserved by calling the section by the previously prepared name .The first line consists of the page header, the second of the page anchor and the side menu and the third consists of the page tail...and the distance between the elements is set to 60 pixels.

Step 3:-

The interior items are placed in the main menu in a horizontal position

CSS

```
grid-template-columns: repeat(4, 1fr)
```

Step 4: -

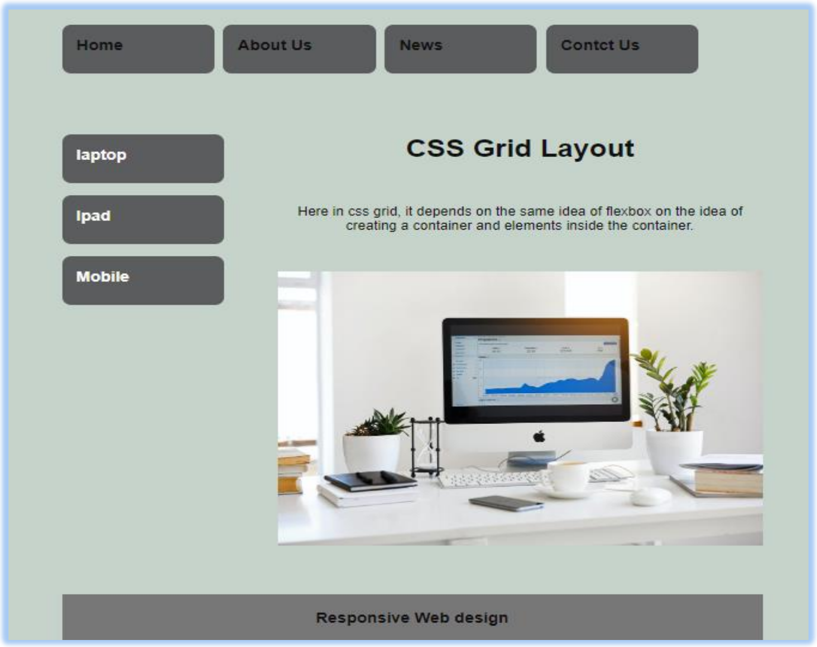
The side menu is divided into one column and the elements are placed in a vertical position.

CSS

```
grid-template-columns: repeat(1, 2fr)
```

This means that he then divided the section into one column and rows by the number of elements in one column.

The result is as follows:-



CSS Grid Layout

Here in css grid, it depends on the same idea of flexbox on the idea of creating a container and elements inside the container.

Figure49 : Shows grid layout for the Desktop

3.5.3 Website design for small screens:-

- **Design using Flexbox Technology**

Design using Flexbox Technology Here the media queries technology is used, which determines the target screen size, and the properties of the technology are used as determined by the developer based on the layout of the desired mode for each screen size.

Flexbox technology has been designed and updated to suit all types of screens through many of the characteristics and values that have been studied in Chapter 3.

Step 1: -

The target screen size is determined by the maximum screen width of 600 pixels, for example, and is executed under this condition, for which the condition does not apply is not executed (true or false).

CSS

```
@media (max-width: 600px)
```

Step 2: -

On small screens, the side menu and the main page are vertical instead of horizontal rows as follows.

CSS

```
 .main-and-sidebar {
       flex-direction: column;
}
```

Step 3:

The elements in the header are repositioned vertically instead of in a row.

CSS

```
 .flex-container {
       flex-direction: column;
     }
```

Step 4: -

The items in the side menu become a row instead of a column**.**

CSS

```
.flex-sidebar {               flex-direction: row;        height: 50PX;          }
```

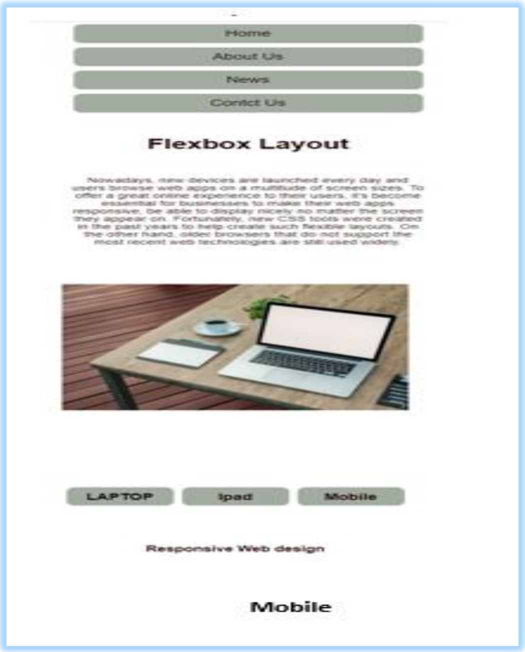
Step 5: -

Finally, one can design the images and make them responsive with the following code:

CSS

```
.main img {    width: 100%;}
```

The result is as follows:-



Nowadays, new devices are launched every day and users browse web apps on a multitude of screen sizes. To offer a great online experience to their users, it's become essential for businesses to make their web apps responsive, be able to display nicely no matter the screen they appear on. Fortunately, new CSS tools were created in the past years to help create such flexible layouts. On the other hand, older browsers that do not support the most recent web technologies are still used widely.

Figure 50: flexbox layout for Mobile

- **Design using Grid Technology**

Design using grid technology here I also used the media queries technology, which determines the target screen size, and then the characteristics of the technology are used as determined by the developer based on the layout of the desired mode for each screen size.

Step 1

Media Query technology was used to determine the target screen size.

CSS

```
@media (max-width: 700px)
```

Step 2- :

Divide the main container instead of two columns into one column, and this is to summon the names and arrange them as follows.

CSS

```
.container{
          grid-template-columns: 1fr;
               grid-template-areas:
          "header"
          "main"
         "sidebar"
        "footer";  }
```

Step 3:-

Also, re-dividing even the header of the page instead of the four columns into one column and arranging them as follows.

CSS

```
.cards1 { grid-template-columns: repeat(1, 1fr); }
```

Step 4:-

The side menu is divided into 3 columns instead of a column.

CSS

```css
.cards { grid-template-columns: repeat(3, 1fr); }
```

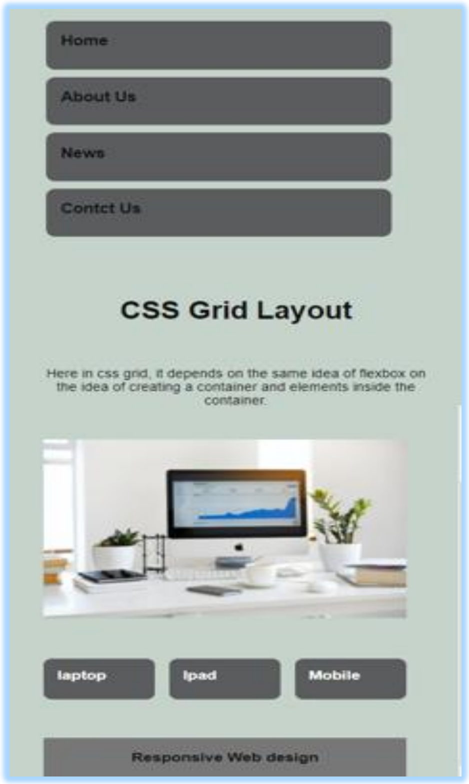
Step 5: -

Finally, design the images and make them responsive with the following code.

CSS

```css
.main img {    width: 100%;}
```

The result is as follows:-



CSS Grid Layout

Here in css grid, it depends on the same idea of flexbox on the idea of creating a container and elements inside the container.

Figure 50: Shows grid layout for Mobile

3.6 Summary and Conclusion

Grid and flexbox technologies are characterized by flexibility, strength, and many features. As verified in this chapter, the developer has options of all types and it is up to him/her to determine what and how to use them.

This chapter provides the developer with the most appropriate choices the site is usually divided into several almost standard sections: a header, a footer, a header page, and a side menu.
The Grid technology was easier to divide (previously considered complicated) by calling the name of each section in the required place so that the division is very clear and easy.

In flexbox technology, one needs to create a nested container for elements to place them in the right place. This is complicated and may result in overlapping elements.

For header creation, flexbox technology has more flexibility. As the goal is to place items in a row for large screens and in a column for small screens, it is simply using the code "width: flex", without knowing the number of elements.

Changing the orientation in small screens was automatic, because this technique has flexibility, and the elements alone fill the (flexible) void. While in grid technology, it was necessary to know the number of elements, i.e. to know the number of columns to plan to place them on small screens, and the matter is somewhat complicated, especially when adding or canceling an element.

This point is mentioned in different items of the strategy: "planning first or content first".

4 The Responsive Web Application Design (RWAD) Model

In this chapter, a responsive web design model called **RWAD** is proposed and implemented using a keen combination of CSS Flex-and-CSS Grid technologies. This model adapts the advantages and avoids the disadvantages of both technologies as concluded in chapter 3. The goal is to blend their greatest features and create an effective responsive layout.

As mentioned previously, integration of these two technologies has been done before, especially in the well-known package of Bootstrap (latest version 5) with the major disadvantage that it is not possible to correct or maintain the site without restarting the whole process including the reload of the huge amount of bootstrap codes [7]. This has been explained in Chapter 1.

The CSS Grid and the CSS Flexbox techniques are integrated here without Bootstrap and with the ability to determine and correct errors and make updates without restarting the whole process or reloading non-needed codes. This of course saves time and space and provides an efficient implementation of responsive sites.

Since the implementation is a key to success in RWD [3], a fully-fledged RWAD model is developed and presented in an algorithmic implementation mode. To make it a guide-like model, the RWAD is exemplified by a “General Service” website with diversified content to suit a wide range of requirements.

The general responsive website to be exemplified here is made so standard application of general services of a company aimed to promote its services and reach target audiences having different screen sizes.

The site consists of a main page divided into two sections, a section with an introductory video to introduce a company and another section with a text welcoming visitors. Also, the “About us” section has a creation date and a detailed look at the company and its goals. The services section is divided into three sections, corresponding to the assumed three main services provided by the company. For example, Land, Sea, and Air transportation services. Another section contains the company's customers by displaying their logos, a section contains the company's partners, a section displays the company's responsible persons, and another section for communicating with the company's customers.

4.1 RWAD Implementation Algorithm

This is a step-by-step method showing how to develop an RWD site using an advantageous integration of the powerful CSS Grid and Flexbox. Steps 5 and step 11 of this method show that it is fairly simple to determine and correct errors and enhance the content, while it would be so complicated in **Bootstrap** where there is no access to the source code and therefore the whole site must be reloaded. all source codes are available for those who are interested to test or use them inside the University of Tripoli.

Step1:- Create a new file as follows:

Figure51 : Shows the creation of a file

Step2:- Create a new file as follows (vs studio code)

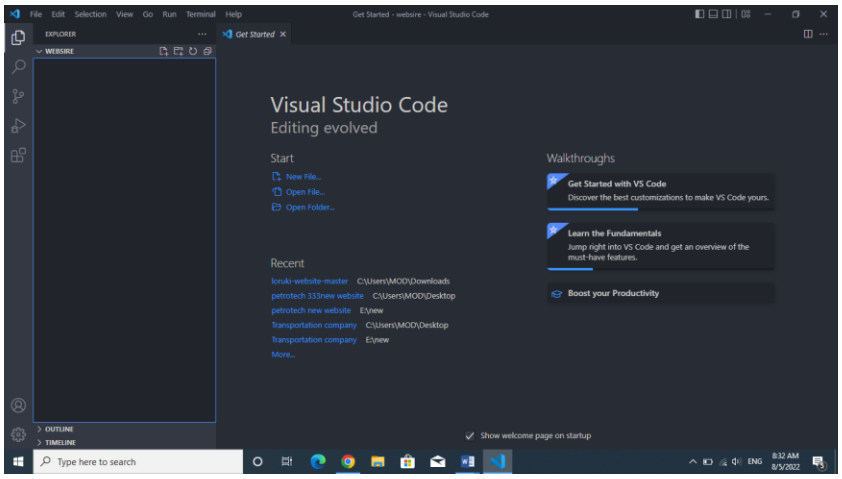
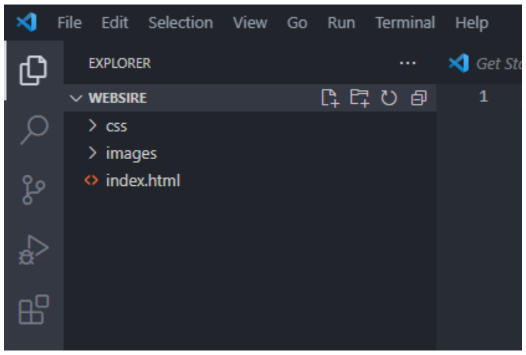

File
Edit
Selection
View
Go
Run
Terminal
Help
Get Started - websire - Visual Studio Code
EXPLORER
WEBSIRE
Get Started
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Recent
loruki-website-master C:\Users\MOD\Downloads
petrotech 333new website C:\Users\MOD\Desktop
petrotech new website E:\new
Transportation company C:\Users\MOD\Desktop
Transportation company E:\new
More...
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.
Boost your Productivity
Show welcome page on startup
OUTLINE
TIMELINE
Type here to search
ENG
8:32 AM
8/5/2022

Figure 52:The Microsoft script editor explains

Step3:- Create site-specific files

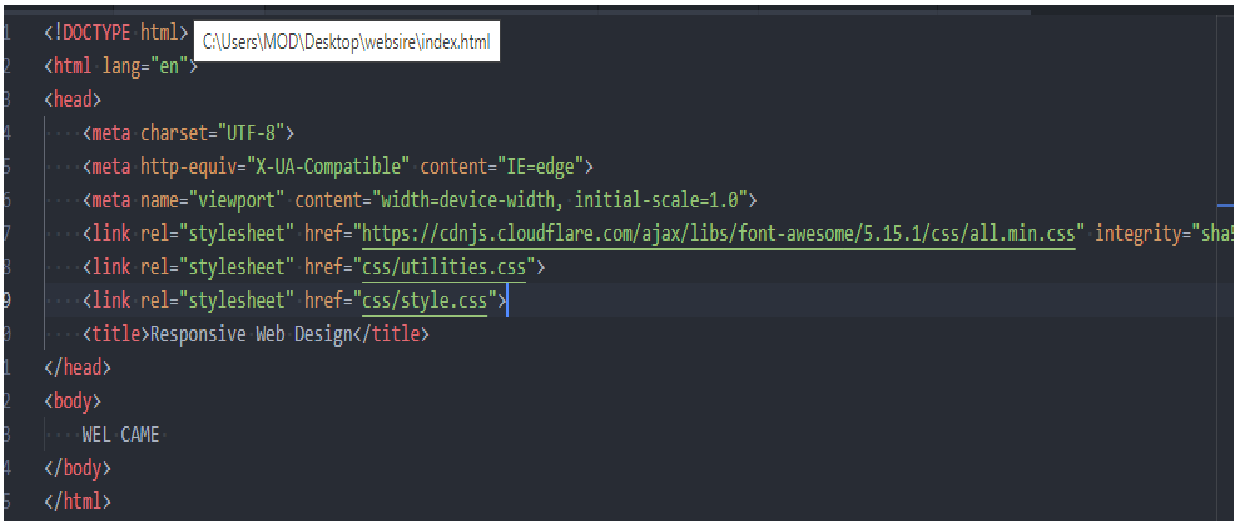
Create a file (index.html) for the home page of the site, create another folder for CSS design, and create another folder for images as follows: -



Step4:- Create a home page

On the main page, the title of the site has been added, as well as a call to the design files, the call to the font type library, any library, and any functions to be called on this page.

HTML



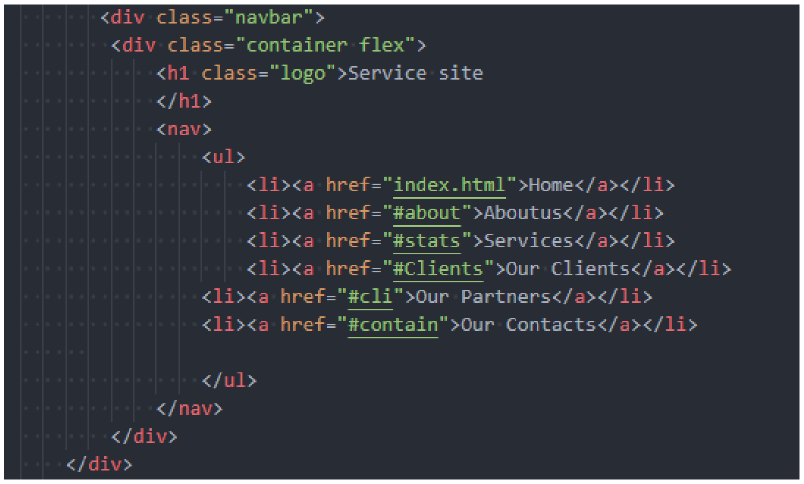
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css" integrity="sha
    <link rel="stylesheet" href="css/utilities.css">
    <link rel="stylesheet" href="css/style.css">
    <title>Responsive Web Design</title>
</head>
<body>
    WEL CAME
</body>
</html>
```

Step5:- Create Section **(Header)**

This section is a navbar list containing links to the main elements of the site and when there is a list, it is better to use **Flexbox** technology with its flexibility while dealing with elements, as well as it has many easy and flexible features.

In the HTML file, create a container with two elements, because this technique depends on the parent and children system, and the name is given to the container class flex.

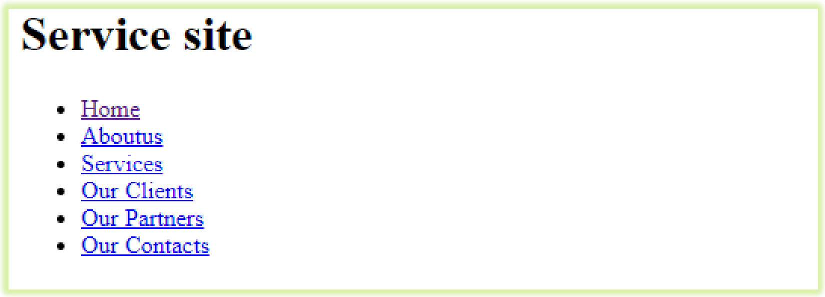
HTML



```
    <div class="navbar">
     <div class="container flex">
         <h1 class="logo">Service site
         </h1>
         <nav>
             <ul>
                 <li><a href="index.html">Home</a></li>
                 <li><a href="#about">Aboutus</a></li>
                 <li><a href="#stats">Services</a></li>
                 <li><a href="#Clients">Our Clients</a></li>
             <li><a href="#cli">Our Partners</a></li>
             <li><a href="#contain">Our Contacts</a></li>

             </ul>
         </nav>
     </div>
 </div>
```

The elements in the HTML file are vertical as in the following figure:-



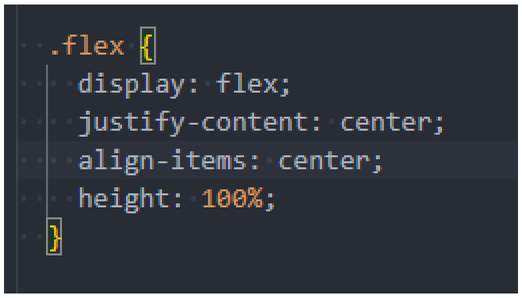
Service site

- Home
- Aboutus
- Services
- Our Clients
- Our Partners
- Our Contacts

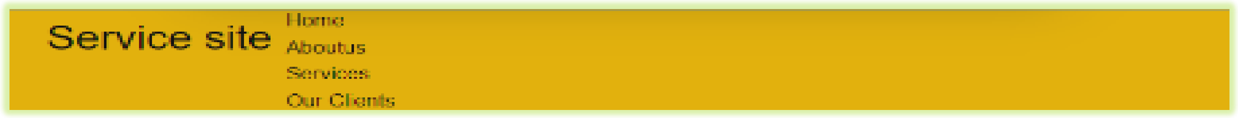
Figure 53: Shows the implementation for the page header

Elements should be placed horizontally with the heading on the left and the elements on the right as is known on any website .At the design stage, create a Category **(.flex)** that has many characteristics that have already been studied in this thesis **Chapter 3**. Display mode: flex to take advantage of the characteristics of this technique, the default orientation of the elements is also row to make the elements in the middle of the container and an equal distance between the elements using the following properties: justification-content: center and alignment-elements: Center.

CSS.



```
.flex {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
}
```



The internal elements of the list must be horizontal, and the following code is used:

CSS

```
.navbar ul {
  display: flex;
}
```

The result will be as in the following figure:-

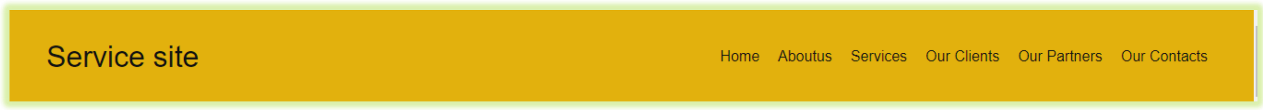


Figure 54: The implementation for the page header2

To make this section responsive to all screen sizes and different browsers, the size of the target screens must be determined using media query technology, which is one of the basics used in responsive design and is the key to determining when to change the interface and apply the new rules specified in media queries.

CSS

```
@media (max-width: 800px)
```

Here, in the previous code, the targeted screens are of medium size such as the iPad screens, and the code under this condition is executed. For the elements inside the header, the orientation is changed to a vertical position to fit the size of the screens.

CSS

```
@media (max-width: 800px) {
.navbar .flex {
  flex-direction: column;}
}
```

The result will be as in the following figure:-

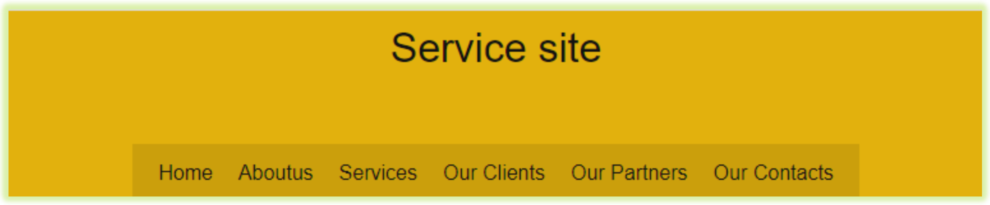


Figure 55:Shows the implementation for the page header3

This is a good display using a medium-sized screen, but when displayed on a smaller screen, an error occurs an undesirable band pops up as follows:

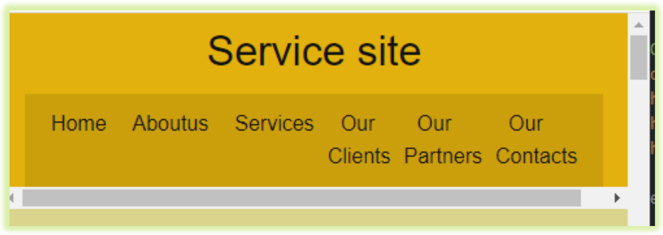


Figure 56 .The error displays the execution in the size of small screens

If this kind of error happens with **Bootstrap**, it will not be possible to determine the breakpoint where the error occurred to correct it. Therefore one is forced to reload the whole site with all unnecessary codes of bootstrap including those used for other purposes.

In **RWAD,** it is a much easier task: As all breakpoints are programmable, to make the header adapt to smaller screens such as mobile phone screens, one can test the elements in a specific screen size, and if the scroll bar appears, one just adds a specific breakpoint and elements will be rearranged by the flex technology according to the properties' values provided. This is shown in the following code where a new breakpoint is added:-

CSS

```
@media (min-width: 600px)
 ul{
  flex-direction: column;
 }
```

The result will be as in the following figure:-

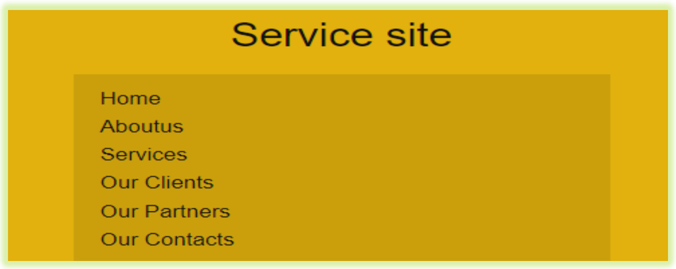


Figure:57 Shows the implementation for the page header4

In the previous code, the elements were changed from a row to a column at a breakpoint of 600 pixels, but when the site was run at a breakpoint of 400 pixels, the same problem occurred even after the elements were in a vertical position.

The result will be as in the following figure:-

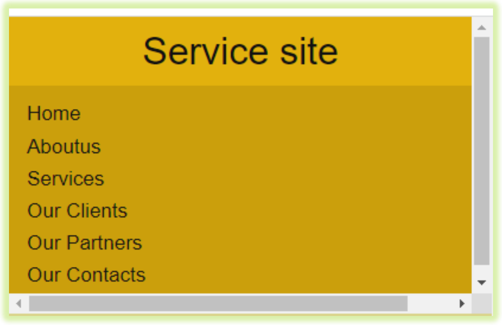
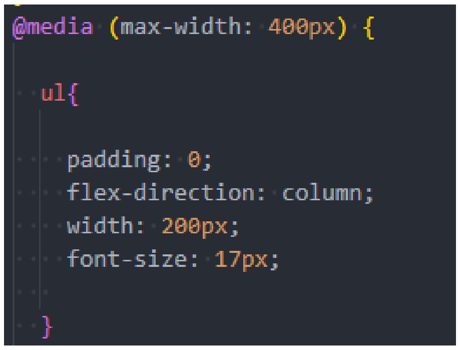


Figure 58 .The screen width at a breakpoint is 400 pixels

To solve this problem, a new breakpoint is created, which is the point where the error occurred, and at that point, and changed the width of the container to a smaller width that is compatible with the size of the screen. Also, change the font size to a smaller size.



```
@media (max-width: 400px) {

  ul{

    padding: 0;
    flex-direction: column;
    width: 200px;
    font-size: 17px;

  }
```

The result will be as in the following figure:-

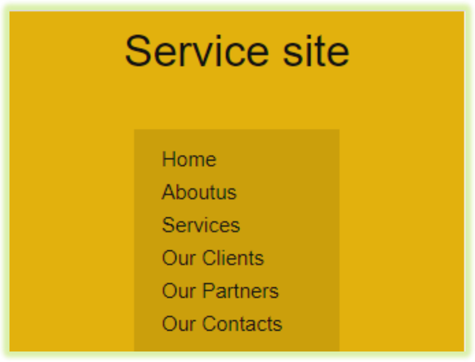


Figure 59:The screen width at a breakpoint is 400 pixels2

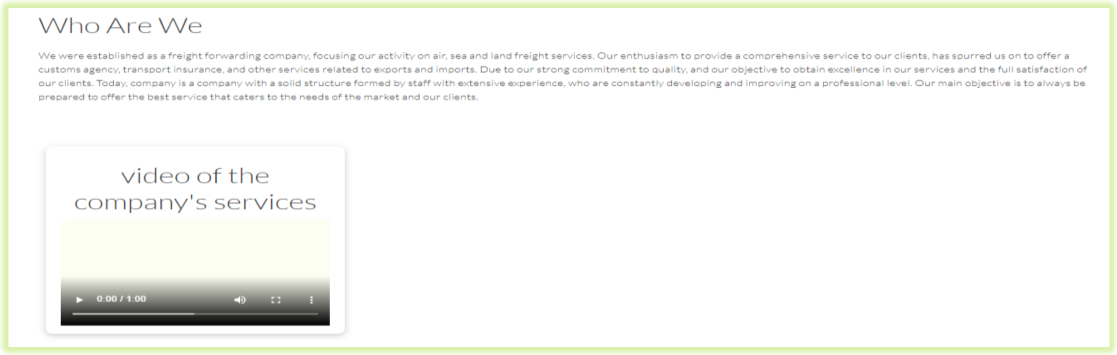
Step6:- Create (Home page)

In this section, a container was created containing two elements: the main element contains the text of an introduction to a company, and the other element contains an introductory video about the company as follows: -

HTML:-

```
<section class="showcase">
    <div class="container grid">
      <div class="showcase-text">
        <h1>Who Are We   </h1>
        <p>We were established as a
              </div>
      <div class="showcase-form card">
                        <div class="title text-center">
        <h2>video of the company's services </h2>
        <div class="border"></div>
       <video controls>
        <source src="1.mp4" type="video/mp4">
       </video>
     </div>
          </div>
  </section>
```

In HTML, the elements are vertical as follows:-


Who Are We

We were established as a freight forwarding company, focusing our activity on air, sea and land freight services. Our enthusiasm to provide a comprehensive service to our clients, has spurred us on to offer a customs agency, transport insurance, and other services related to exports and imports. Due to our strong commitment to quality, and our objective to obtain excellence in our services and the full satisfaction of our clients. Today, company is a company with a solid structure formed by staff with extensive experience, who are constantly developing and improving on a professional level. Our main objective is to always be prepared to offer the best service that caters to the needs of the market and our clients.

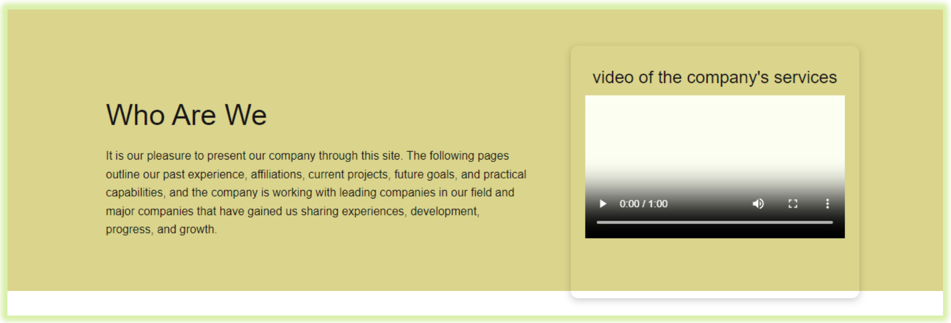
video of the company's services

Figure 60 :demonstrates the implementation of the home page

This section contains two main elements. It is better to use the **grid** technique because this technique depends on the pre-planning approach, and it has many characteristics that make it a powerful technique. This is done by creating (**class. Grid**) and dividing the section into two parts with the same space of 1fr, with the distance between the elements equal to 20px, and with the elements in the middle of the container.

CSS

```
.grid {
display: grid;
 grid-template-columns: repeat(2, 1fr);
  gap: 20px;
 justify-content: center;
align-items: center;}
```


Who Are We

It is our pleasure to present our company through this site. The following pages outline our past experience, affiliations, current projects, future goals, and practical capabilities, and the company is working with leading companies in our field and major companies that have gained us sharing experiences, development, progress, and growth.

video of the company's services

Figure 61:demonstrates the implementation of the home page2

To make this section responsive, the following breakpoints have been added for medium screens.

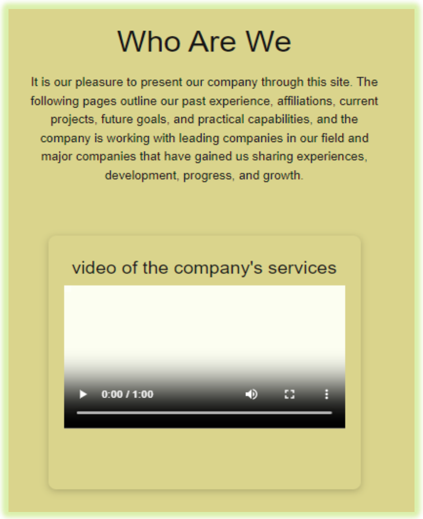
CSS

```
@media (max-width: 1100px) {
```

Re-layout the section to one column only so that items in small screens are vertical.

CSS

```
.grid {  grid-template-columns: 1fr;}}
```



Who Are We

It is our pleasure to present our company through this site. The following pages outline our past experience, affiliations, current projects, future goals, and practical capabilities, and the company is working with leading companies in our field and major companies that have gained us sharing experiences, development, progress, and growth.

Figure 62:demonstrates the implementation of the home page3

After experiments, the problem that the Form is not responding to small screens was discovered

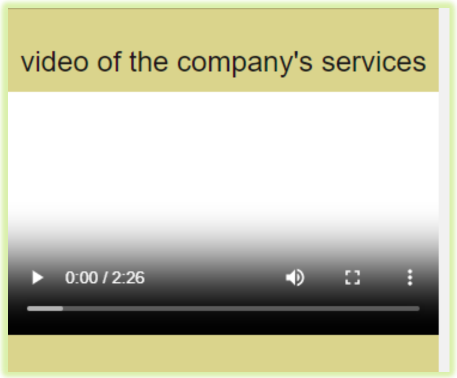


Figure 63:Displays the form size

To solve this problem, the size of the form at the breakpoints has been changed to 400 pixels and 600 pixels as follows:

```
@media (max-width: 600px) {
showcase-form {
    width: 370px;
  }
```

```
@media (max-width: 400px) {
.showcase-form {
    width: 300px;
  }
```

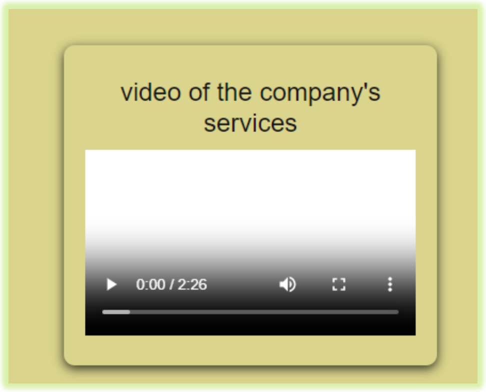
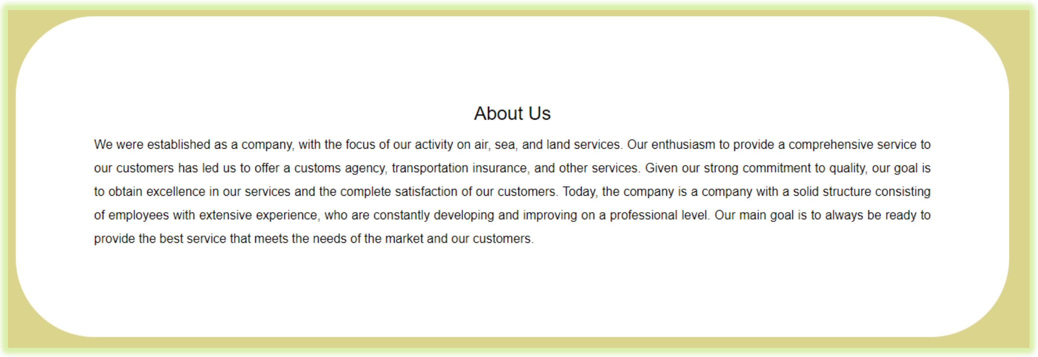


Figure 64:Displays the form size2

Step7:- Create (About Us)

This section usually contains one part as an overview of the company, so there is no need in the design stage to use a grid or flexbox.



About Us

We were established as a company, with the focus of our activity on air, sea, and land services. Our enthusiasm to provide a comprehensive service to our customers has led us to offer a customs agency, transportation insurance, and other services. Given our strong commitment to quality, our goal is to obtain excellence in our services and the complete satisfaction of our customers. Today, the company is a company with a solid structure consisting of employees with extensive experience, who are constantly developing and improving on a professional level. Our main goal is to always be ready to provide the best service that meets the needs of the market and our customers.

Figure65 :the implementation of the page shows information about us

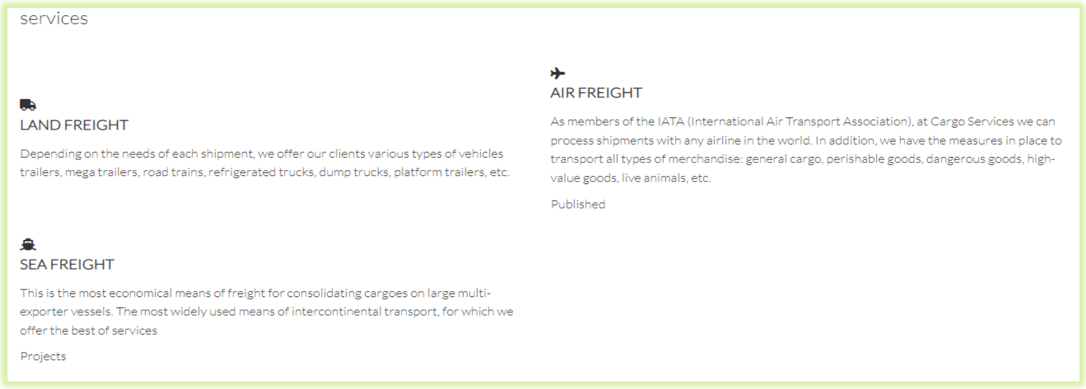
Step8:- Create Section (Services Section)

The Services Department contains three main elements of a land services company, naval services, and air services. Create a container with three elements, like this:

Html

```
 <section class="stats">
       <div class="container">
           <h1 class="stats-heading text-center my-1">
               services
           </h1>
           <div class="grid grid-3 text-center my-4">
               <div>
                   <i class="fa fa-truck"></i>
                   <h3 class="content-image__text__title">LAND FREIGHT</h3>
                   <p>Depending on the needs of each shipment, we offer our clients various types of vehicles
trailers, mega
                     trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.</p>
                         </div>
               <div>
                   <i class="fa fa-plane"></i>
                   <h3>AIR FREIGHT </h3>
                   <p>As members of the IATA (International Air Transport Association), at Cargo Services we
can process
                     shipments with any airline in the world. In addition, we have the measures in place to
transport all types
                     of merchandise: general cargo, perishable goods, dangerous goods, high-value goods, live
animals, etc.</p>
                   <p class="text-secondary">Published</p>
               </div>
               <div>
                   < i >class="fa fa-ship"></i>
                   <h3>SEA FREIGHT</h3>
                   <p>This is the most economical means of freight for consolidating cargoes on large multi-
exporter vessels. The
                     most widely used means of intercontinental transport, for which we offer the best of
services</p>
                   <p class="text-secondary">Projects</p>
               </div>
           </div>
       </div>
   </section>
```

The vertical elements are as follows:


services

LAND FREIGHT

Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.

AIR FREIGHT

As members of the IATA (International Air Transport Association), at Cargo Services we can process shipments with any airline in the world. In addition, we have the measures in place to transport all types of merchandise: general cargo, perishable goods, dangerous goods, high-value goods, live animals, etc.

Published

SEA FREIGHT

This is the most economical means of freight for consolidating cargoes on large multi-exporter vessels. The most widely used means of intercontinental transport, for which we offer the best of services

Projects

Figure 66: Demonstrates the implementation of the services page

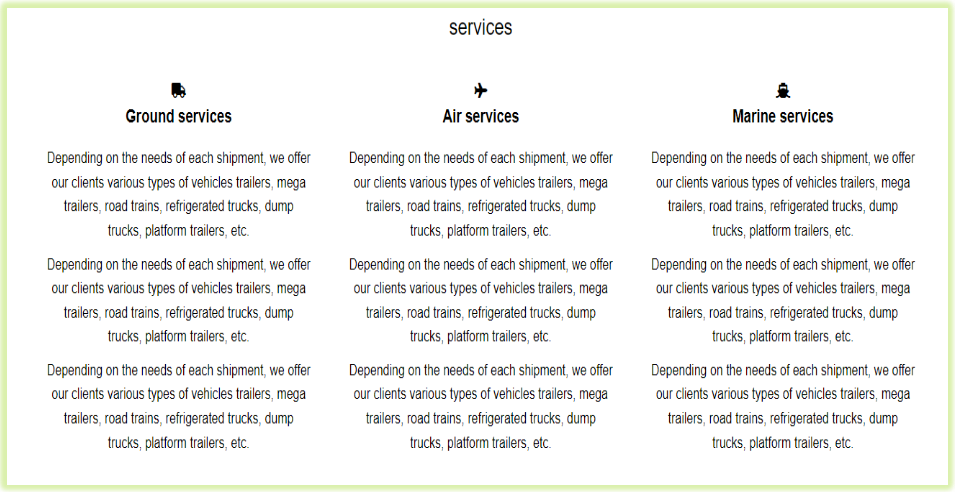
It is known that there are three elements, and it is unlikely to add any element later, so the pre-planning **grid** technique can be used.

CSS

```
.grid-3 {

 grid-template-columns: repeat(3, 1fr);

}
```


services

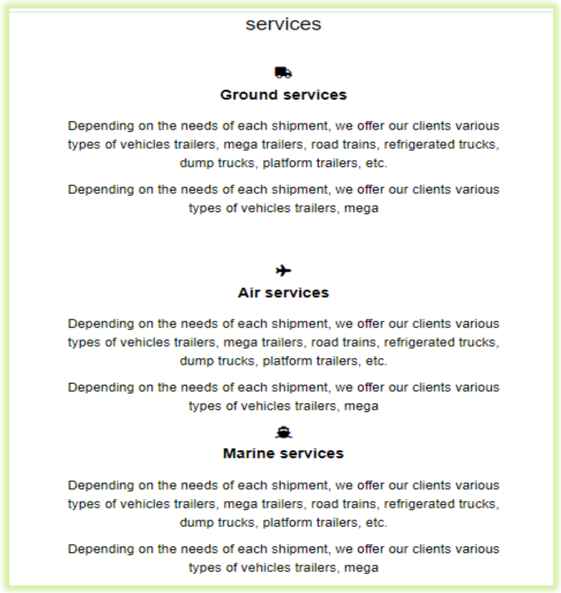
Ground services	Air services	Marine services
Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.	Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.	Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.
Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.	Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.	Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.
Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.	Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.	Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.

Figure 67:Demonstrates the implementation of the services on page2

Also, in screens with a small size, the division must be changed to one column according to a specific breakpoint according to the test for the partition to be responsive.

CSS

```
grid-template-columns: 1fr;
```


services

Ground services

Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.

Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega

Air services

Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.

Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega

Marine services

Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega trailers, road trains, refrigerated trucks, dump trucks, platform trailers, etc.

Depending on the needs of each shipment, we offer our clients various types of vehicles trailers, mega

Figure:68 Demonstrates the implementation of the services on page3

Step9:- Create Section (Company Clients)

A section for customer companies is presented with their logos, which contain many elements.

Create a container with items as follows:-

HTML

```
<section class="Clients">
 <h3 class="md text-center my-2">
  <P>Our Clients</P>
 </h3>
 <div class="container flex">
    <div class="card">
          <img src="images/logos/logo-1.png" alt="">
    </div>
    <div class="card">
```

```
          <img src="images/logos/logo-2.png" alt="">
        </div>
        <div class="card">

          <img src="images/logos/logo 3.ico"">
        </div>
        <div class="card">

          <img src="images/logos/logo 4.ico" alt="">
        </div>
        <div class="card">
                 <img src="images/logos/logo 5.ico" alt="">
        </div>
        <div class="card">
                <img src="images/logos/logo 6.ico" alt="">
        </div>
        <div class="card">
                   <img src="images/logos/logo 7.ico" alt="">
        </div>
  </div>
</section>
```

During the design process here, it is better to use **flexbox** technology because there are many elements without a specific layout and this technology is flexible, it does not care about the number of elements, it is flexible so that the elements are arranged automatically to fill the available space to fit the screen, plus it is easy to add or cancel any element, and it is flexible in viewing at different screens. The **Flexbox** property created at the beginning can be called by any section as needed.

The result is as follows:

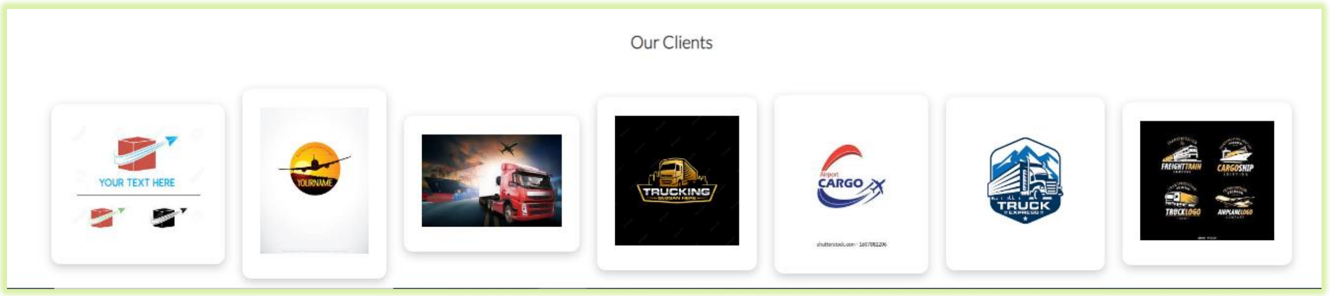


Figure 69 .The implementation of the page shows the company's customers

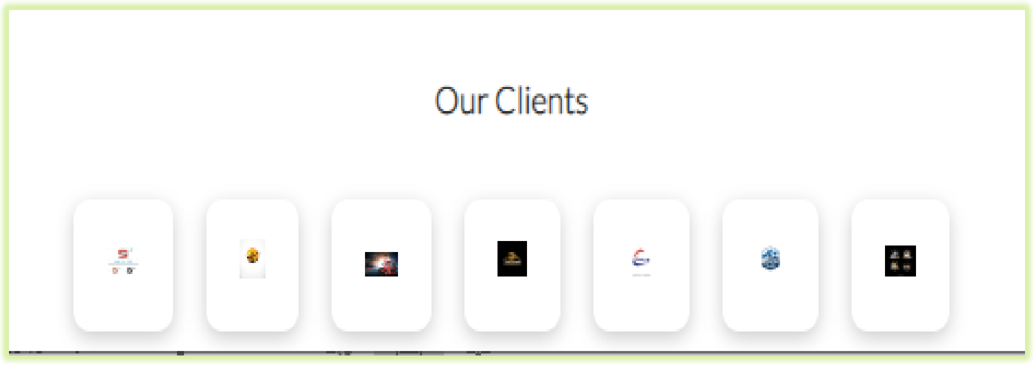


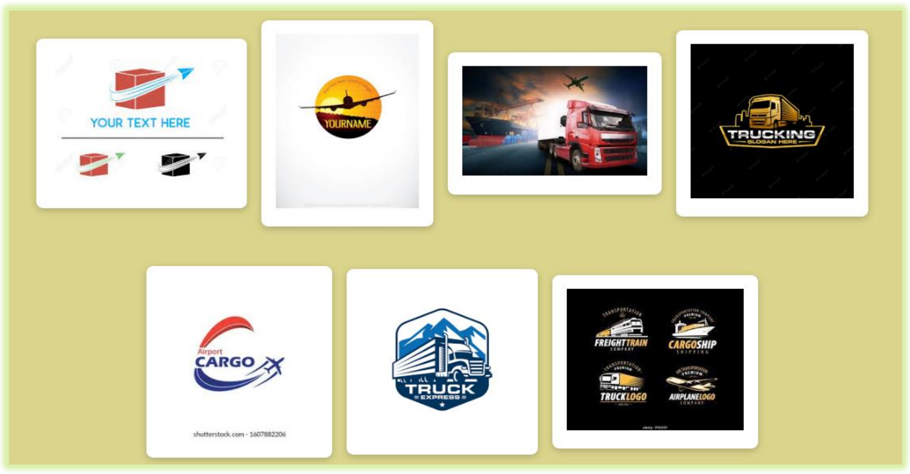
Figure 70 .The implementation of the page shows the company's customers

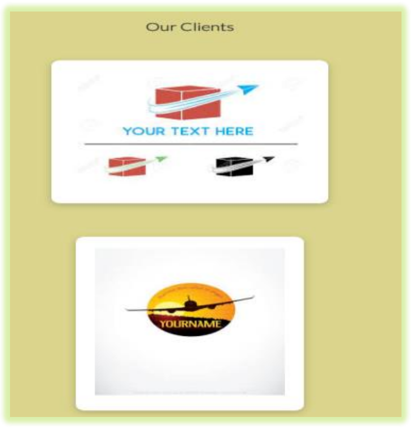
It is shown in the previous two figures, the elements are in one row, but due to the smaller container space, the logos become smaller and unclear, they must be processed .For the elements to be clear and fill the space, the flex-wrap property is used, so that the elements become responsive to the size of the container and are displayed appropriately:

CSS

```
.Clients .flex {  flex-wrap: wrap;}
```

The default mode for this flex-wrap technique is no wrap .If the total width of the elements is greater than the width of the container, it is allowed to respond through this property automatically so that the elements go to fill the next row without specifying their place, they fill and adapt to the remaining space :





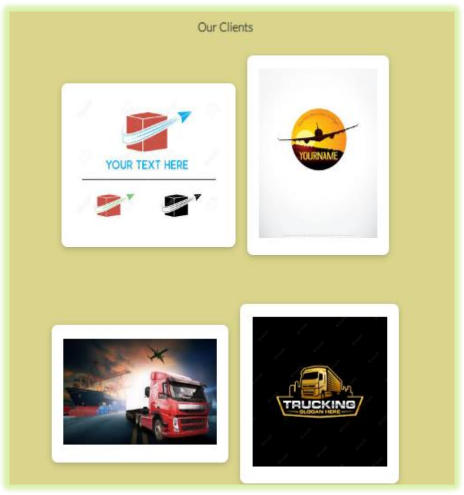


Figure 71The implementation of the page shows the company's customers3

Items respond to any screen size without the need for Media query technology.

Step10:- Create (Company Partners)

In this section, the company's contracting partners are presented by displaying their company logos: - a new file as follows:

Html

```
<section class="cli">
 <h3 class="md text-center my-2">
   <P>Our Partners </P>
  </h3>
       <div class="container grid">
         <img src="images/cli logo/1.png" alt="">

           <div class="card">
             <img src="images/cli logo/2.jpg" alt="">
           </div>
           <div class="card">
             <img src="images/cli logo/3.jpg" alt="">
           </div>
           <div class="card">
             <img src="images/cli logo/4.jpg" alt="">
           </div>
```

```
<div class="card">
  <img src="images/cli logo/5.jpg" alt="">
</div>
<div class="card">
  <img src="images/cli logo/6.webp" alt="">
</div>
<div class="card">
  <img src="images/cli logo/7.jpg" alt="">
</div>
<div class="card">
  <img src="images/cli logo/8.png" alt="">
</div>
<div class="card">
  <img src="images/cli logo/9.png" alt="">
</div>
<div class="card">
  <img src="images/cli logo/10.jpg" alt="">
</div>
<div class="card">
  <img src="images/cli logo/11.jpg" alt="">
</div>
<div class="card">
  <img src="images/cli logo/12.jpg" alt="">
</div>
<div class="card">
  <img src="images/cli logo/14.jpg" alt="">
</div>
<div class="card">
  <img src="images/cli logo/15.png" alt="">
</div>
            </section>
```

In this section, there are many elements, with a condition to display the largest partner (larger than the other elements.).It means that there is a plan to put the elements, automatically exclude the use of Flexbox **and use Grid technology** because it is the planning technique first and depends on the rows and it is easy to put the elements in any suitable place for them, integrate and control it when a specific idea.

In the previous HTML code, the(Class grid) was called, which was then configured in the previous sections to take advantage of its properties, but this Class divides the section into two columns and due to the presence of many elements here, it is required to re-divide the section into 5 predefined columns, unlike the previous section, because after experiment and tests, it was discovered that every 5 images in a row are the clearest thing, as well as to achieve the condition of the largest size of element 1, it must take the space of elements 1 and 2.

CSS

```
cli .grid {
  grid-template-columns: repeat(5, 1fr);
}
.cli .grid > *:first-child {
  grid-row: 1 / span 2;
}
```

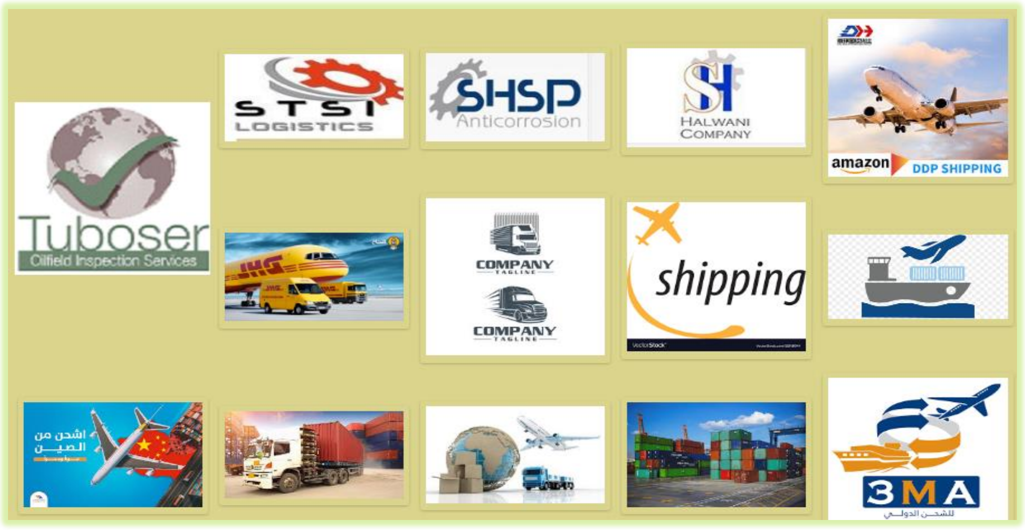


Figure :72 Shows the implementation of the Company Partners page.

For this section to be responsive on small sizes screens, it should be divided into two columns as follows :

CSS

```
.cli .grid {  grid-template-columns: repeat(2, 1fr);}
```

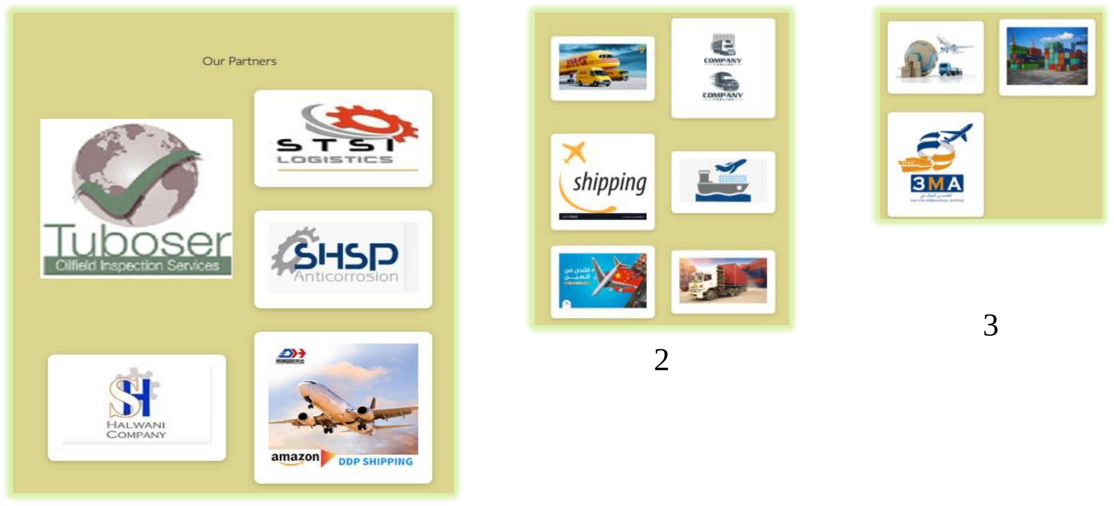


Figure 73 :Shows the implementation of the Company Partners page2

Step11:- Create Section (Contact Us)

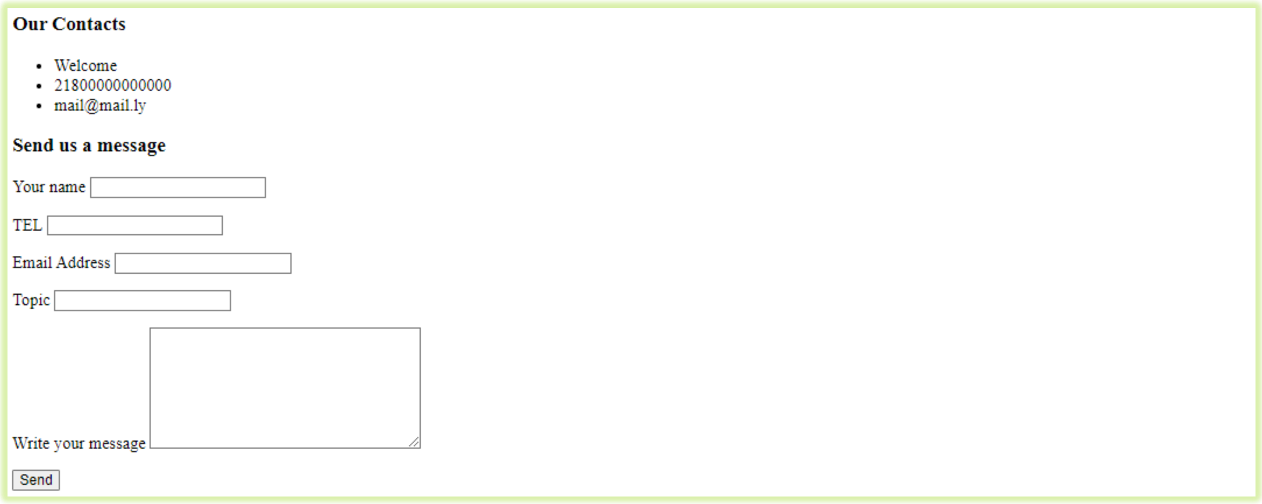
This section is common and known on all sites. The customer communication section contains elements, most of which are text boxes, and it has a specific layout according to the developer's desire. Create a container with the items:-

HTML

```html
<div class="contain" id="contain">
    <div class="wrapper">
      <div class="contacts">
        <h3>Our Contacts</h3>
            <ul>
          <li>WelCome</li>
          <li>21800000000000</li>
          <li>mail@mail.ly</li>
        </ul>
      </div>
          <div class="form">
        <h3>Send us a message</h3>
        <form action="">
          <p>
            <label for="">Your name</label>
```

```
  <input type="text">
</p>
<p>
  <label for="">TEL</label>
  <input type="text">
</p>
<p>
  <label for="">Email Address</label>
  <input type="text">
</p>
<p>
  <label for="">Topic</label>
  <input type="text">
</p>
<p class="full-width">
  <label for="">Write your message</label>
  <textarea name="" id="" cols="30" rows="7"></textarea>
</p>
<p class="full-width">
  <button>Send</button>          </p>          </form>
```

All elements vertical in HTML code



Our Contacts

- Welcome
- 21800000000000
- mail@mail.ly

Send us a message

Your name

TEL

Email Address

Topic

Write your message

Figure 74 .Shows the implementation for the Contact Us page

This was started in the design process and prior planning for the department, and since the elements are of various sizes, it is better to use **Grid** technology. In large screens, with a minimum width of 700 pixels, the section is divided into two columns.

CSS

```
@media (min-width: 700px) {
 .wrapper {
  display: grid;
  grid-template-columns: 1fr 2fr;
 }
}
```

Also, all the inner elements in (From) are in two columns with the same distance between them

CSS

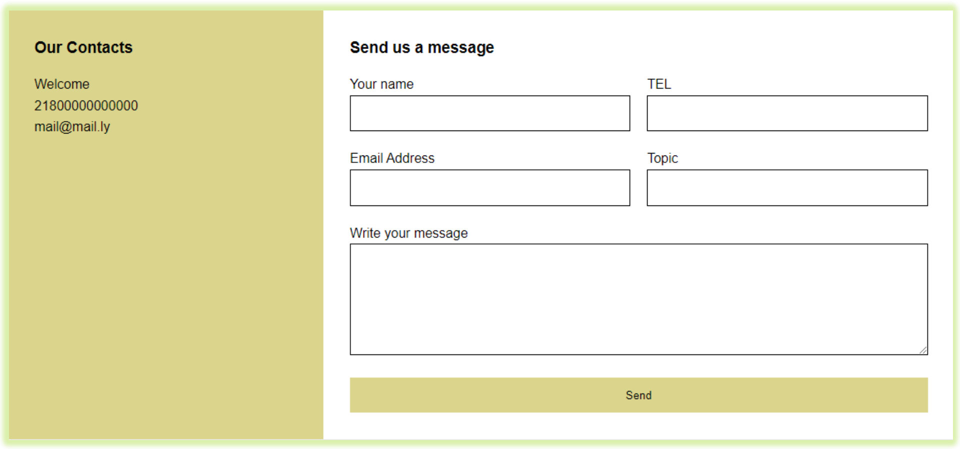
```
form {  display: grid;  grid-template-columns: 1fr 1fr;  grid-gap: 20px;}
```

The width of the last two elements is the width of all the elements after dividing the new one.

CSS

```
.full-width {  grid-column: 1 / 3}
```

The result is as follows:


Our Contacts

Welcome
21800000000000
mail@mail.ly

Send us a message

Figure 75 .Shows the implementation for the Contact Us page2

On small screens, the section is set into one column, so that this section is responsive

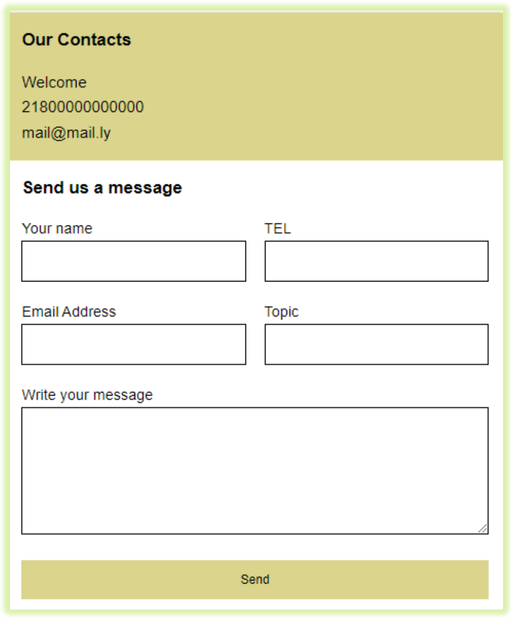


Figure:76 Shows the implementation for the Contact Us page3

It is very easy to change the shape of the textbox on a large screen by setting it, and it can be changed whenever needed. for example:-

Merge textbox1 full row and textbox2 full row as well as textbox3, Text Box 4, and textbox 5 On large screens, they appear as follows:-

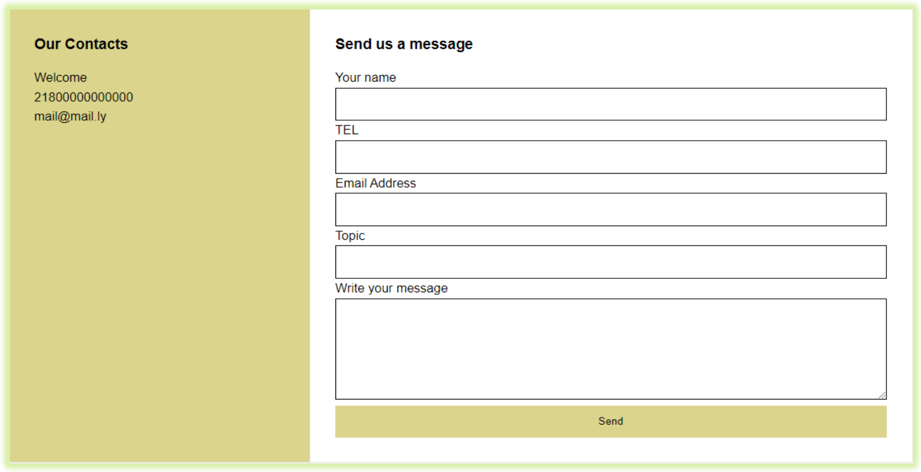


Figure:77 Shows the implementation of the Contact Us page for large-size screens

This is very easy because the location of the code is well-known and specified. the requirement can be changed as follows:

CSS

```
@media (max-width: 600px) {
  form {
    display: grid;
    grid-template-columns: 1fr 1fr;
    grid-gap: 20px;
  }
```

In small screens, it is divided into two columns as follows: -

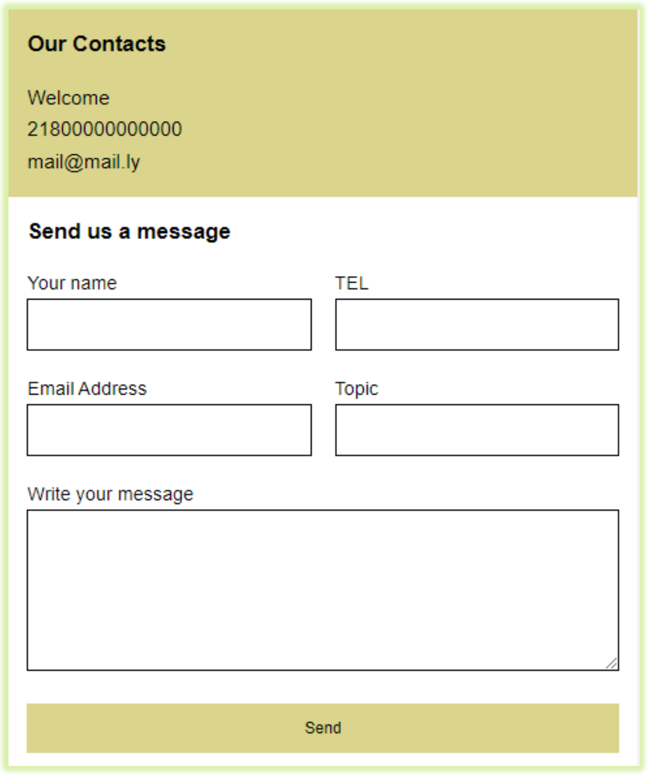


Figure 78.The implementation of the Contact Us page is shown for small-size screens

In **RWAD**, is possible to change it in place because all the breakpoints are known. If this kind of change is required under Bootstrap, it will be very difficult to trace and correct.

Step12:- Create official cards

The purpose of this section is for company officials to display their data for customers to communicate with them. Hence, creating a container as follows: -

Html

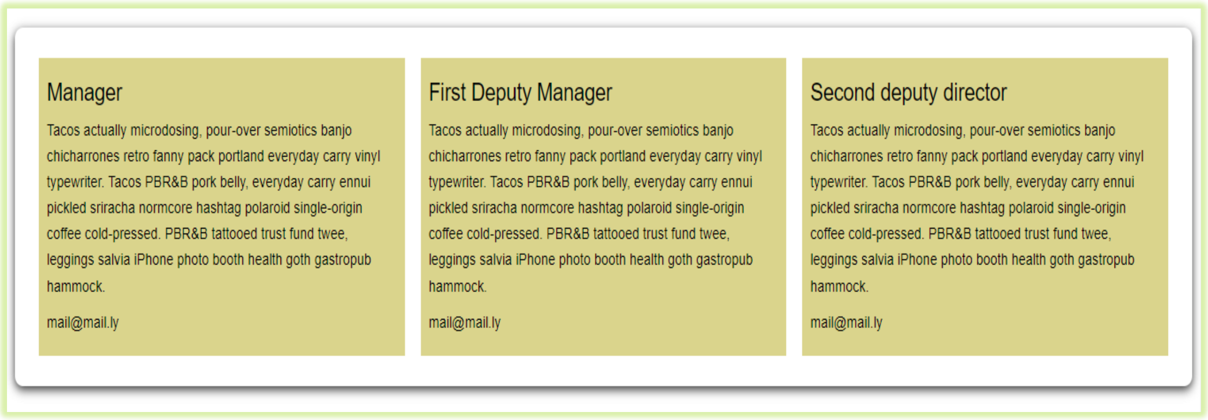
```
<section class="card">
    <article class="card1">
      <h2>Manager</h2>
      <p>Tacos micro dosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday
carry vinyl typewriter. Tacos PBR&B pork
      <p>mail@mail.ly</p>
    </article>
    <article>
      <h2>First Deputy Manager</h2>
      <p>Tacos actually microdosing, pour-over semiotics banjo chicharrones iPhone photo booth health
goth gastropub hammock.</p>
      <p>mail@mail.ly</p>
    </article>
    <article>
      <h2>Second deputy director</h2>
      <p>Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland
everyday carry vinyl typewriter. Tacos PBR&B pork iPhone photo booth health goth gastropub
hammock.</p>
      <p>mail@mail.ly</p>
        </article>
  </section>
```

This section initially contains three items, but these are company officials. One can add an administrator, remove an administrator, or change the order of items. Here, a flexible menu is needed, and it is better to use **Flexbox** technology.

CSS

```
.card{
 display: flex;
}
```

On large screens, items are placed horizontally, which is the default orientation for **flex**



Manager

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

mail@mail.ly

First Deputy Manager

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

mail@mail.ly

Second deputy director

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

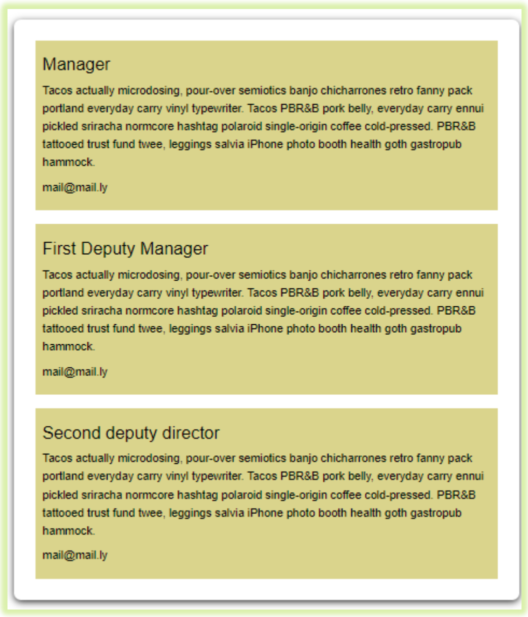
mail@mail.ly

Figure 79 :Displays the implementation for the official cards page

But in small screens, the elements must be vertical, just change the orientation of the elements with the property at a certain breakpoint as per the test

CSS

```
@media (max-width: 600px) {.card{  flex-direction: column;  }}
```



Manager

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

mail@mail.ly

First Deputy Manager

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

mail@mail.ly

Second deputy director

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

mail@mail.ly

Figure80 :Displays the implementation for the official cards page2

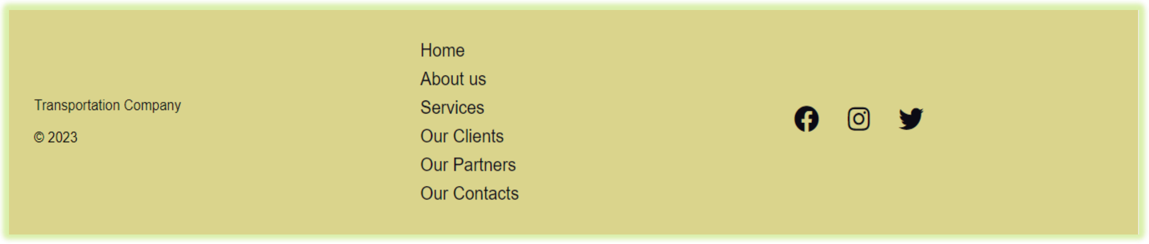
Step13:- Create a section footer

This section consists of three elements that are constructed as follows:-

Html

```
<footer class="footer bg-dark py-5">
     <div class="container grid grid-3">
         <div>
                                    <p>Transportation Company</p> &copy; 2020</p>
         </div>
         <nav>
             <ul>
               <li><a href="index.html">Home</a></li>
               <li><a href="#about">About us</a></li>
               <li><a href="#stats">Services</a></li>
               <li><a href="#Clients">Our Clients</a></li>
           <li><a href="#cli">Our Partners</a></li>
           <li><a href="#contain">Our Contacts</a></li>
             </ul>
         </nav>
         <div class="social">
                         <a href="#"><i class="fab fa-facebook fa-2x"></i></a>
             <a href="#"><i class="fab fa-Instagram fa-2x"></i></a>
             <a href="#"><i class="fab fa-twitter fa-2x"></i></a>
         </div>
     </div>
  </footer>
    </body>
</html>
```

The class **grid** 3 is ready (configured previously) and can be called in the HTML file. the result is as follows:

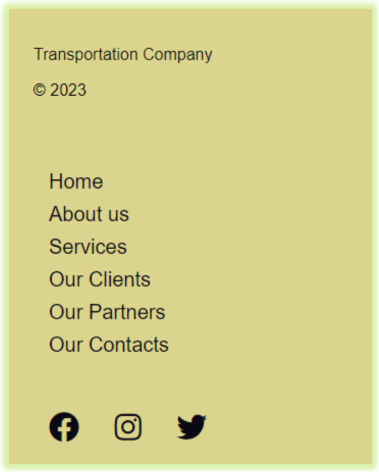

Transportation Company

© 2023

Home
About us
Services
Our Clients
Our Partners
Our Contacts

Figure 81: Displays the implementation for the footer page

On small screens, the result is as follows:


Transportation Company

© 2023

Home
About us
Services
Our Clients
Our Partners
Our Contacts

Figure 82: Displays the implementation for the footer2 page

Step14:- Create a responsive image and video

For the images on the site that are not responsive at the beginning as in the following figure:

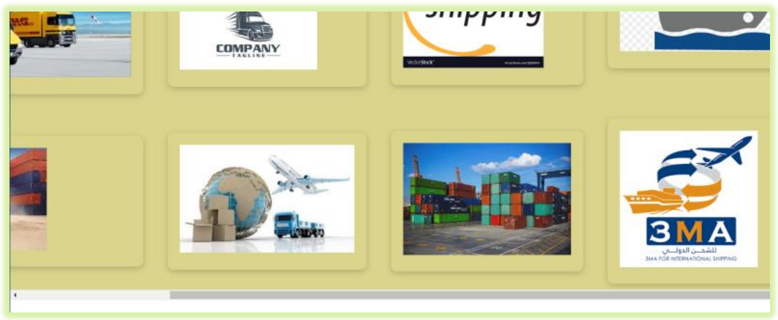


Figure 83:The implementation presents a responsive design for photos and videos

The appearance of the scroll bar did not stop, which resulted in a non-responsive site. This problem can be solved using the technique of flexible images:

CSS

```
img {  width: 100%;}
```

The previous code means if the image is larger than its container, it will be minimized to be the same size as the container. If the image is smaller than the container, the image will be in its virtual size and thus all the pictures on the site become responsive.

And also the video

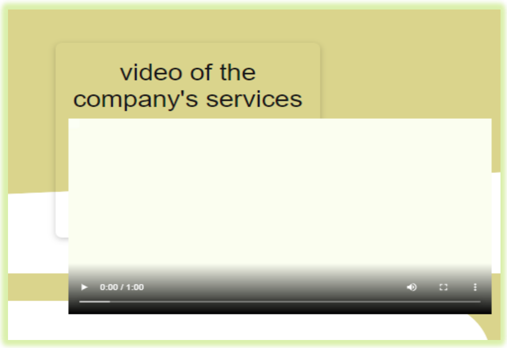


Figure 84.The implementation presents a responsive design for photos and videos2

Here, in the previous figure, the video is larger than the container in which it is located. This problem is solved as follows:

CSS

```
video{  max-width: 100%;  max-height: 100%;}
```

The same code is used to display the video on the same container on all the screen sizes, so the result will be as follows:

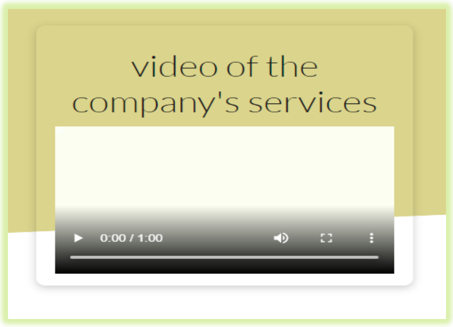


Figure 85 .The implementation presents a responsive design for photos and videos3

The video has become responsive.

4.2 Summary and Conclusion

The CSS flex and Grid technologies are integrated and implemented in one model which is practically illustrated by developing a general site exploiting the advantages and flexibility of each technology as required in each part of development. The site model is general and comprehensive with all kinds of sections and diversified content types so that it can be followed to develop any other RWD site.

This template-like model is developed without using complex responsive design tools such as the widely used bootstrap where one cannot correct any error except by reloading the whole thing and restarting the whole process. This is because Bootstrap is an external tool with a large amount of code which are not available to the developer.

In the proposed RWAD model, any enhancement or error correction can be done directly on the local code of the relevant part without restarting the design or reloading any codes.

The model developed in this thesis contains only the codes required for developing the required site, unlike the external tools that require an excessive amount of unnecessary hidden codes. This saves lots of memory, provides extra speed, and can be operated and maintained efficiently.

5 Conclusion and Recommendation

5.1 Conclusion

The thesis has deeply studied and analyzed the latest two modern RWD technologies, namely **CSS Flexbox and CSS grid**, and made theoretical and practical comparisons showing their similarities, differences, advantages, and disadvantages. Stress was made on methods of their implementation which is a key factor to evaluate any technology according to most studies in the literature survey.

As no software technology can be efficiently implemented without appropriate software tools, the thesis scanned the latest RWD tools, and unlike other surveys, software tools were surveyed in the thesis separately from the survey of technology. This has avoided several ambiguities in previous surveys. Moreover, the technologies themselves were classified in the thesis as basic, old, and modern techniques, which will facilitate any future study.

Based on the analysis of the two mentioned and modern technologies (CSS Grid and Flexbox), the thesis proposed a full-fledged guide model for responsive web application design, named RWAD, which combines the advantages and discards the disadvantages of the two mentioned technologies. The RWAD model is presented algorithmically in an implementation mode using diversified content that may be required by any website application so that it can be used as an effective guideline. Moreover, RWAD does not use any external package like "Bootstrap". This kind of package usually causes high operational overhead, results in a lack of understanding of the underlying programs for the developer, and makes error correction, debugging, and further enhancement very difficult. The RWAD model not only provides clarity to the developer to build an RWD site but also saves computer time and space since it does not require loading huge external codes that may not be even needed for each application. RWAD is also shown to facilitate debugging and site enhancement as source code is directly available for each section.

5.2 Recommendations for further research

Further research studies in the same area may be proposed here to include but not to be limited to the following topics:

- A detailed test of performance on the proposed RWAD model as it was indicted in this thesis to be a very promising model.

- Practical applications of the RWAD model to emphasize effectiveness in more details. This can be achieved by real implementation of reponsive web sites in service areas of transportation, communication, customer relations (e.g. banking, procurement, etc.), and university students' academic affairs. These kind of applications to test RWAD effectiveness are recommended for BSc. graduation projects.

- A study on Media Queries technology, including the development of breakpoints and their various uses in this field.

- Devolopment of GRID technology without using Media Queries.

- A study on responsive typography (changing font sizes) within media queries and/or using viewport units to reflect smaller or larger amounts of screen space.

- A detailed study on the webflow tool, as it is reported to be promising when used with flexbox technology, and may be with grid technology too.

- A detailed development of a combined flexbox and grid model using webflow tools.

- A study of the idea of "Responsive images" because it is reported to be a very promising techniques in RWD field.

5.3 General recommendations

As websites that are not responsive are being widely rejected by an increasing number of institutions, it is very necessary to promote RWD in Libya. It is also highly recommended to introduce the RWD subject within the web programming curriculum in computer sciences and information technology and direct projects in this area.

Any development institution especially the local ones or any researcher in the area of RWD may use this thesis as a learning document, as a development and implementation guide, or as a starting point for further research.

References

[1]. Hristov, H. T., Chochev, N., & Hristov, H. (2022). DESIGN TECHNIQUES AND PRACTICES OF GRID LAYOUTS AND CONTENT OF WEB PAGES. XX. https://www.researchgate.net/publication/360181044.

[2].CSS layout: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout. [viewed 04.03. 2023].

[3] Responsive design: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design#using_media_queries_for_responsive_typography. [viewed `15.11. 2022].

[4]. Giurgiu, L., & Gligorea, I. (2017, July). Responsive web design techniques. In International conference knowledge-based organization (Vol. 23, No. 3, pp. 37-42).

[5]. Bootstrap 5 (Grid system) May 05, 2021: https://getbootstrap.com/docs/5.0/layout/grid/, [viewed 28.02. 2023].

[6].Bootstrap:https://www.arab-reviewer.com/how-bootstrap-helps-to-accelerate-the-web-development/, [viewed 28.02. 2023].

[7]. The 10 Most Common Bootstrap Mistakes That Developers Make: https://www.toptal.com/twitter-bootstrap/the-10-most-common-bootstrap-mistakes, [viewed 28.02. 2023].

[8]. 6 Most Common Bootstrap Mistakes to Avoid in Web Development: https://www.geeksforgeeks.org/6-most-common-bootstrap-mistakes-to-avoid-in-web-development/, [viewed 04.04. 2023].

[9]. Author Lawrence Othman, Publisher: lorens Osman, 2020, (The Depths In CSS Grid)

[10]. Zhang, Y. (2020). C-RWD: A Computational Responsive Web Design Service.

[11]. Top 6 Mobile App Development Trends for 2023: https://sharkbyte.ca/top-6-mobile-app-development-trends-for-2023/ [viewed 04.06. 2023].

[12] Desktop vs Mobile Market Share Worldwide - September 2023 https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide#yearly-2011-2023

[13.] WHAT PERCENTAGE OF INTERNET TRAFFIC IS MOBILE?: https://www.oberlo.com/statistics/mobile-internet-traffic,[viewed 04.06. 2023].

[14]. Is Mobile-Friendliness A Google Ranking Factor: (October 21, 2022): https://www.searchenginejournal.com/ranking-factors/mobile-friendliness/#close,[viewed `20.12. 2022].

[15]. Is Fluid Layout Better Than Fixed Layout? 19 February, 2023: https://www.hurix.com/fluid-layout-vs-fixed-layout/, [viewed 01.04. 2023].

[16]. *Rogatnev Nikita Responsive Web Design*. (2015).

[17]. Adaptive Design: https://www.interaction-design.org/literature/topics/adaptive-design [viewed 01.04. 2023].

[18] Position:https://developer.mozilla.org/en-US/docs/Web/CSS/position. [viewed 15.11. 2022].

[19] float: https://developer.mozilla.org/en-US/docs/Web/CSS/float.[viewed 15.11. 2022].

[20]. Harb, E., Kapellari, P., Luong, S., & Spot, N. (2011.). Responsive Web Design.

[21] CSS Responsive Image Tutorial: How to Make Images Responsive with CSS: https://www.freecodecamp.org/news/css-responsive-image-tutorial/.[viewed [04.04. 2023].

[22] Beginner's guide to media queries: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Media queries.[viewed `04.04. 2023].

[23]. CSS Grid and Flexbox—What Problems They Solve March 03, 2022: https://www.telerik.com/blogs/grid-flexbox-what-problems-they-solve, [viewed 28.02. 2023].

[24]. When to use Flexbox and when to use CSS Grid, January 6, 2023: https://blog.logrocket.com/css-flexbox-vs-css-grid/?fbclid=IwAR0-iuwQLVef3B29pBD6qfMIEqW9DqJIR0itasJgNsqMGfbScywzaEKPDfE, [viewed 28.02. 2023].

[25].Giurgiu, L., & Gligorea, I. (2017, July). Responsive web design techniques. In International conference knowledge-based organization (Vol. 23, No. 3, pp. 37-42).

[26].Marcotte, Ethan. (2011). Responsive web design. A Book Apart/Jeffrey Zeldman.

[27].Marcotte, E. (2017). Responsive web design: A book apart n 4. Editions Eyrolles.

[28] Frain, Ben. (2012). Responsive web design with HTML5 and CSS3 : learn responsive design using HTML5 and CSS3 to adapt websites to any browser or screen size. Packt Publishing Ltd.

[29] Natda, K. V. (2013). Responsive web design. Eduvantage, 1(1).

[30].Firdaus, T. (2014). Responsive Web Design by Example: Beginner's Guide. Packt Publishing Ltd.

[31] Mohamed, A. A., Cheruiyot, W. K., Rimiru, R., & Ondago, C. (2014). Responsive Web Design inFluid Grid Concept Literature Survey. International Journal of Engineering and Science. Retrieved from http://www. this. com/papers/v3-i7/Version-3 G, 373049057.

[32].Almeida, F., & Monteiro, J. (2017). The Role of Responsive Design in Web Development. Webology, 14(2).

[33]. Emmanuel Ohans (2017). Understanding Flexbox.

[34].Raza, A., Ghazanfar, R., & Raza, M. (2018). RESPONSIVE WEB DESIGN ACCORDING TO RESOLUTIONS.

[35].Bradač, I. (2018). CSS Grid raspored.

[36]. Niess, G., Roubal, A., Thurner, S., & Roque, E. B. (2019). CSS Grid Layouts.

[37]. Suman Aryal (2019.)BOOTSTRAP-A Front-End Framework For Responsive Web Design.

[38].Selonen, J. (2020). Responsiivinen mobiilisivusto CSS-ohjelmistokehyksellä.

[39].Ajitha, P. (2020) Responsive Design in Web Development with Security Using Optimization Algorithms.

[40] Zhang, Y. (2020). C-RWD: A Computational Responsive Web Design Service.

[41].Stoeva, M. (2021). Evolution of Website Layout Techniques. In Computer Technologies and Applications-Anniversary International Scientific Conference, September 15-17, 2021, Pamporovo, Bulgaria.

[42].Regmi, S. (2022). Building a responsive website.

[43]. Bootstrap (front-end framework): https://harmash.com/posts/what-is-bootstrap-and-how-to-use-it, [viewed 01.06. 2023].

[44]. Webflow:https://colorlib.com/wp/responsive-web-design-tools/ [viewed 01.06. 2023].

[45]. what is Adobe Edge Inspect? : https://subscription.packtpub.com/book/web-development/9781849694001/1/ch01lvl1sec03/so-what-is-adobe-edge-inspect, [viewed 06.01. 2023].

[46]. Adobe Edge: https://helpx.adobe.com/mena_ar/dreamweaver/using/preview-webpage-edge-inspect.html, [viewed 07.01. 2023].

[47]. Adaptive Images Deliver small images to small devices: https://adaptive-images.com/ [viewed 07.01. 2023].

[48] Mobile-Friendly Test (by Google): https://colorlib.com/wp/responsive-web-design-tools/viewed 08.01. 2023].

[49]. Mobile-Friendly Test: https://support.google.com/webmasters/answer/6352293?hl=en[viewed 08.01. 2023].

[50]. FitVids: https://support.google.com/webmasters/answer/6352293?hl=en,[viewed 08.01. 2023].

[51]. FitVids: http://fittextjs.com/ [viewed 08.01. 2023].

[52]. One Cloud for all device and browser testing needs.: https://smartbear.com/product/bitbar/ [viewed 08.01. 2023].

[53]. Golden grid system: https://goldengridsystem.com/, [viewed 08.01. 2023].

[54]. https://www.uxpin.com/: https://crozdesk.com/software/uxpin [viewed 08.01. 2023].

[55]. CSS Flexible Box Layout: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout,[viewed 05.02.2023].

[56]. A Complete Guide to Flexbox (Updated on Dec 9, 2022): https://css-tricks.com/snippets/css/a-guide-to-flexbox/,[viewed 05.02.2023]

[57]. Flexbox: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox, [viewed 05.02.2023].

[58].CSS, Flexbox Gap: https://learn.coderslang.com/0046-css-flexbox-gap/, [viewed 4.11.2023].

[59]. CSS Grid Layout: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout,[viewed 05.02.2023].

[60]. A Complete Guide to CSS Grid (Updated on Feb 2, 2023): https://css-tricks.com/snippets/css/complete-guide-grid/,[viewed 05.02.2023]

[61]. CSS Grid Layout: The Fr Unit Published on September 3, 2020, · Updated on April 4, 2022: https://www.digitalocean.com/community/tutorials/css-css-grid-layout-fr-unit,[viewed 05.02.2023]

[62]. grid-area:https://developer.mozilla.org/en-US/docs/Web/CSS/grid-area?retiredLocale=ar, [viewed 05.02.2023].

[63]. Quick! What's the Difference Between Flexbox and Grid? (Updated on Feb 14, 2019: https://css-tricks.com/quick-whats-the-difference-between-flexbox-and-grid/?fbclid=IwAR3P1WLM0fqb49GXBa581DShbKEb9LAmlRhJovC72p2vlonJUrbmf2va3Y4 , [viewed 05.02.2023].

[64]. Difference between CSS Grid and Flexbox: https://blog.knoldus.com/difference-between-css-grid-and-flexbox/?fbclid=IwAR3X2RwW3Z_pVZTWoeRLg7bmwElxNHDvteVm_CfVfqU0VlxrA_nkblygYK4, [viewed 05.02.2023].

[65]. Grid vs Flexbox: Which Should You: Choosehttps://www.webdesignerdepot.com/2018/09/grid-vs-flexbox-which-should-you-choose/?fbclid=IwAR2g7AI9x5Vm9EITnyHpmD5Bq1TfattFH41mphRQRDy-Qcn-8ijhZBdC8Jo, [viewed 04.12.2023].

[66]. CSS GRID VS FLEXBOX:https://x-team.com/blog/css-grid-vs-flexbox/?fbclid=IwAR1JUHZNxZlV9seX10i8eD4CZNDMF9xC7NNl5bEzDI7o3K9t3bTBj6m5PmI /,[viewed 05.02.2023].

الملخص

تصميم الويب سريع الاستجابة **(RWD)** هو منهج تطوير مواقع الويب يتكيف تلقائيا مع المتصفحات المختلفة وأحجام شاشات المستخدمين سواء كانت شاشة حاسب مكتبي كبيرة أو هاتفا ذكيا صغيرا. تتناول هذه الأطروحة هذا المجال الجديد من حيث تقنياته البرمجية وطرق تنفيذها.

تقوم الأطروحة بمسح وعرض الأدبيات المتعلقة بالتقنيات والأدوات البرمجية الشائعة المستخدمة بفعالية في التصميم المتجاوب **RWD**. وعلى خلاف الاستطلاعات الأخرى، يميز الاستطلاع في هذه الأطروحة بين أدوات البرمجة وبين تقنيات البرمجة المستخدمة في **RWD**. وتؤكد الادبيات أن التركيز الحالي في هذا المجال ينصب على تقنيتين هامتين وأكثر فعالية من غيرهما في مجال التصميم المتجاوب **RWD**، وهما المعروفتان باسم **CSS Grid** و **CSS Flexbox [1، 2، 3]** ، وبالتالي تقدم الأطروحة تحليلات ومقارنات عميقة لهاتين التقنيتين تظهر ميزات وعيوب كل منهما بالنسبة للأخرى.

بناء على هذه المقارنات، تقترح الأطروحة نموذجا يسمى RWAD (تصميم تطبيقات الويب سريع الاستجابة). يجمع RWAD بين التقنيتين المذكورتين أعلاه باعتماد مزاياهما وتجاهل عيوبهما. نظرا لأن التنفيذ يعد نجاحا رئيسيا في أي تقنيات RWD [4] ، فقد تم صياغة نموذج RWAD المقترح خوارزميا بنمط تنفيذي باستخدام موقع RWD كامل مع محتوى متنوع بحيث يمكن استخدامه كنموذج إرشادي.

تم الجمع بين هاتين التقنيتين من قبل، خاصة في الحزمة واسعة الاستخدام والمعروفة باسم **Bootstrap**5 **[5،6]** في أحدث إصداراتها. ولكن من الصعب للغاية ، إن لم يكن من المستحيل في هذه الحزمة، تصحيح أي أخطاء في الموقع أو إجراء أي تحسينات ما لم يتم إعادة التشغيل من البداية وإعادة تحميل كمية هائلة من برامج مكتبة الحزمة **[7 ، 8].**

في النموذج RWAD ، يتم دمج تقنيات **CSS Grid** و**Flexbox CSS** دون استخدام **Bootstrap** أوغيره، ومع القدرة على تصحيح الموقع وصيانته وتحديثه دون إعادة تشغيل الموقع برمته ولا إعادة تحميل أي برامج. هذا بالطبع يوفر وقت المعالجة وسعة التخزين مع فعالية تنفيذ المواقع المتجاوبة.

جامعة طرابلس
كليـة العلوم
قسم الحاسب الآلي

تصـميم التطبيقـــات الويـــب سريـعة الاسـتجابة

محمد احمد إمحمد الأعور

مصطفي فاتح عبدالعال
بروفيسور

قدمت هده الرسالة استكمالا لمتطلبات الإجازة العالية(الماجستير) في علوم الحاسب الالي

بتاريخ 9 ربيع الآخر, 1445 هـ الموافق 24 /2023/10م