

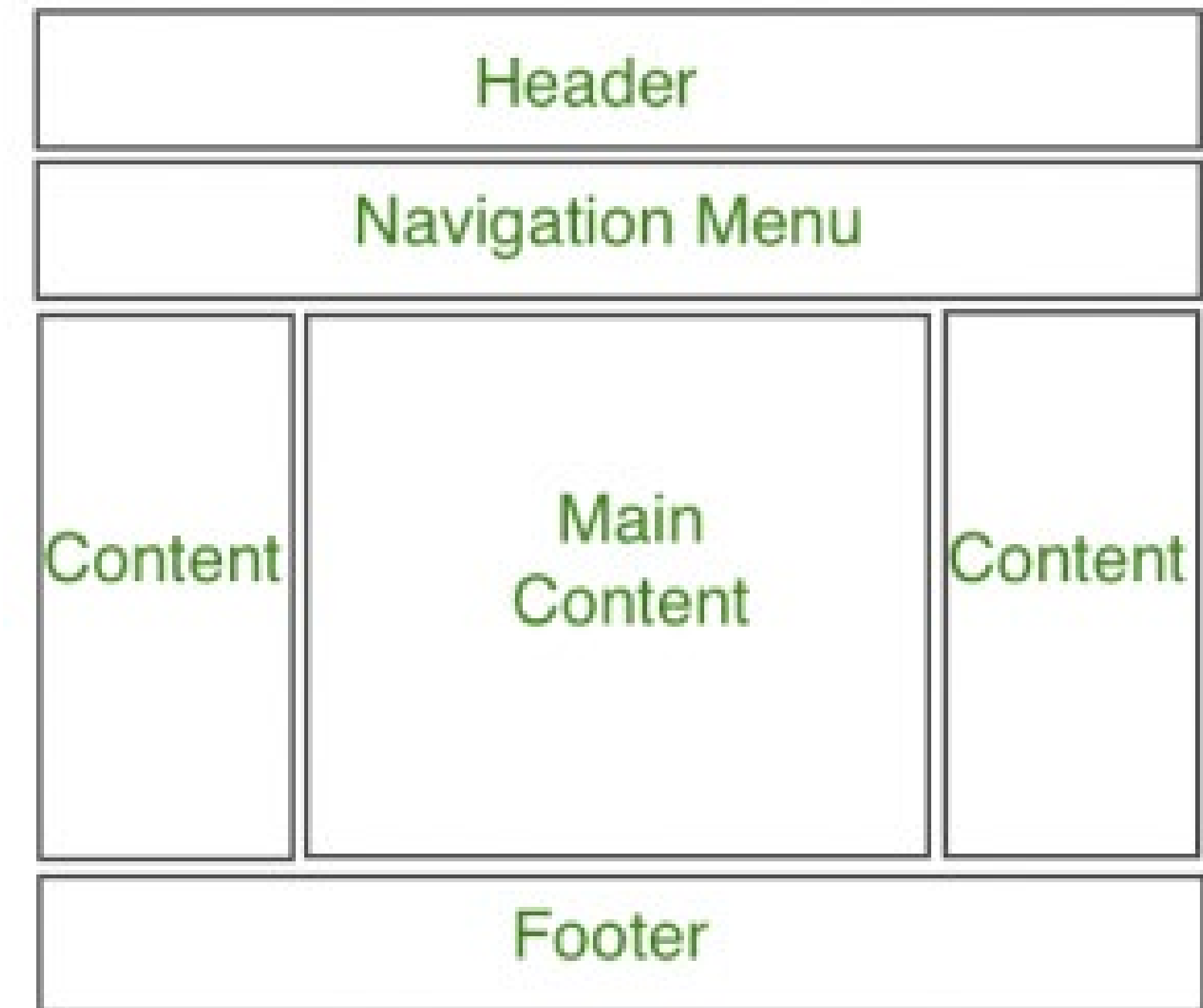
# CASCADING STYLE SHEETS : CSS

## 02 LAYOUT GRID and FLEXBOX

INFO 2302 WEB TECHNOLOGIES

# What is Web Layout...?

[https://developer.mozilla.org/en-US/docs/Learn/Common\\_questions/Design\\_and\\_accessibility/Common\\_web\\_layouts](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Design_and_accessibility/Common_web_layouts)







## Main content A

## Stuff on the side B1/B2

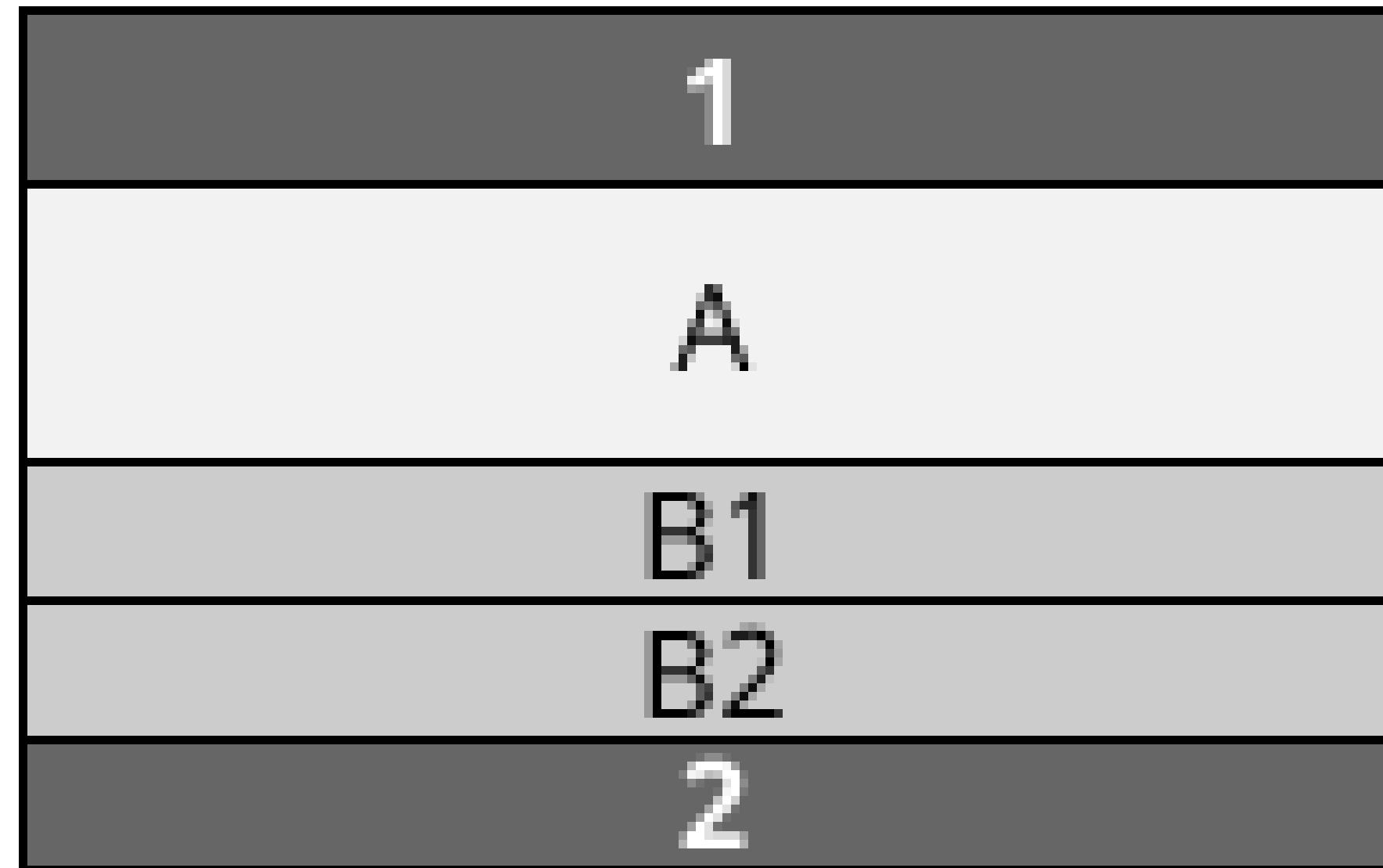
1) Information complementing the main content; 2) information shared among a subset of pages; 3) alternative navigation system. In fact, everything not absolutely required by the page's main content.

## Footer 2

Visible at the bottom of every page on the site. Like the header, contains less prominent global information like legal notices or contact info.

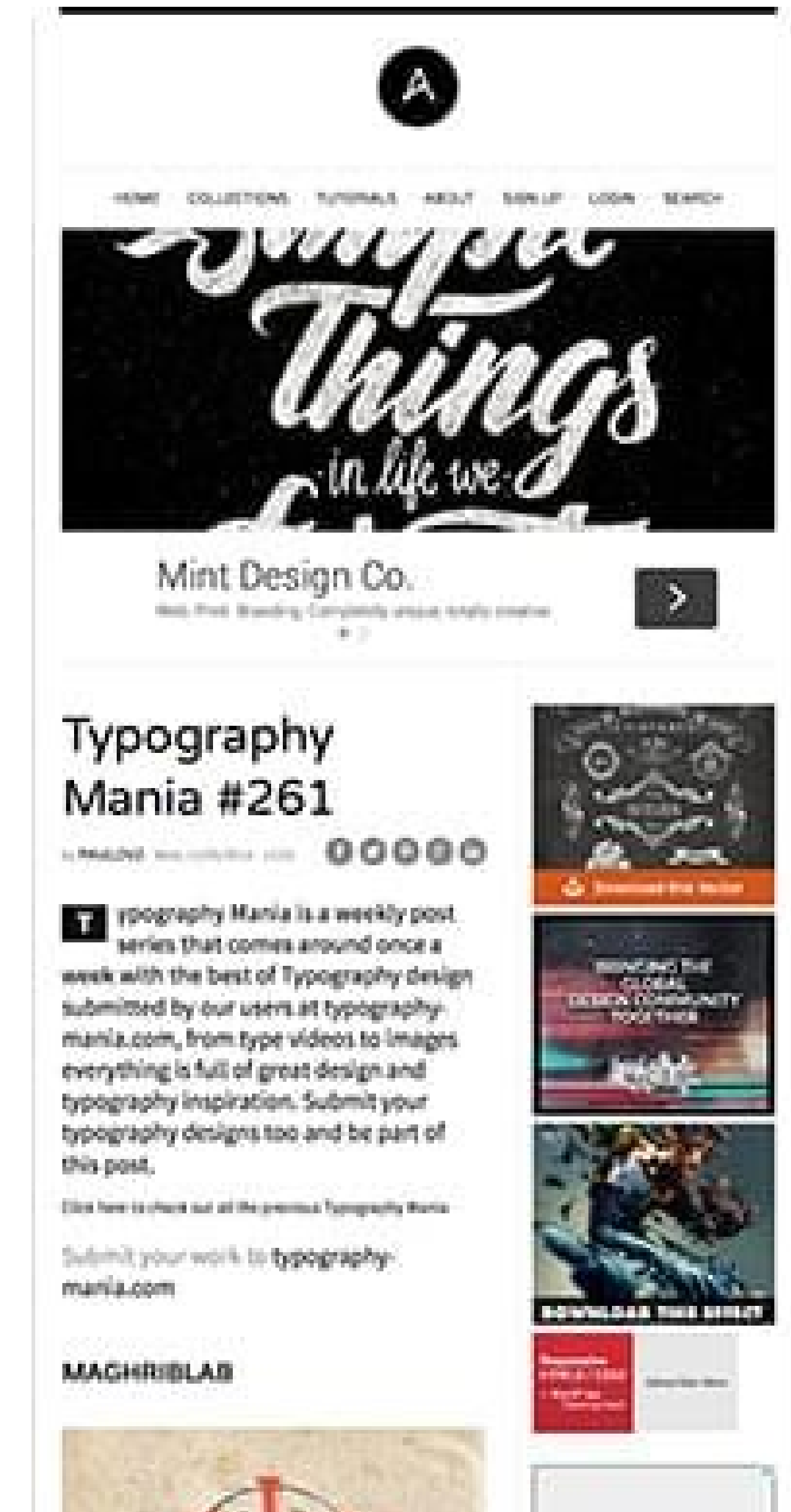
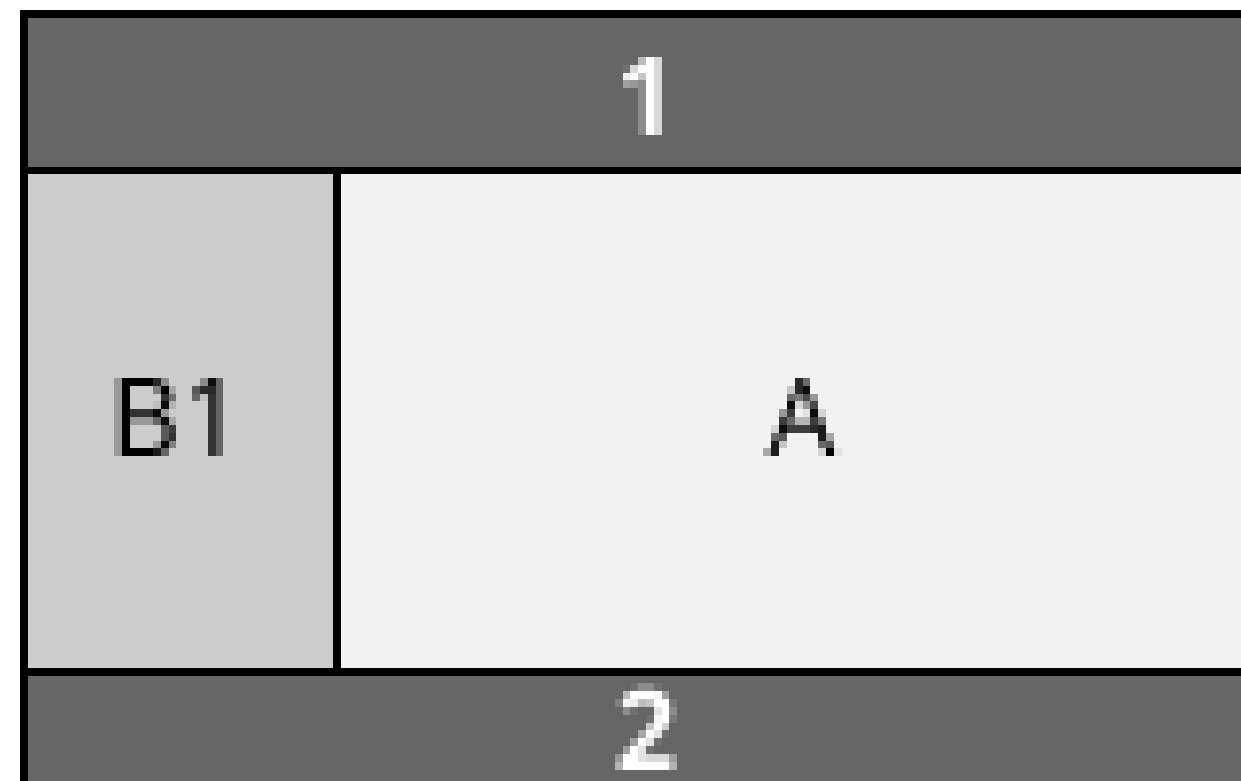
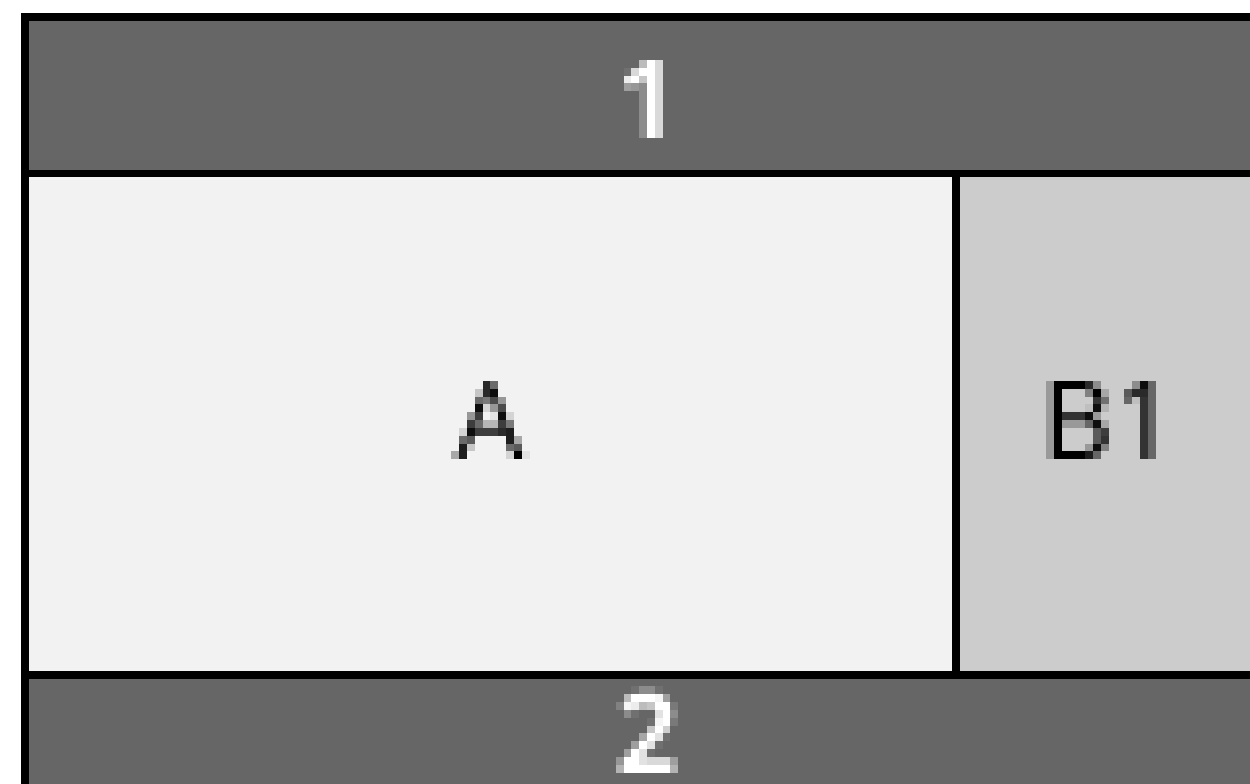
# • COLUMN LAYOUT – 1 column

**1-column layout.** Especially important for mobile browsers so you don't clutter the small screen up



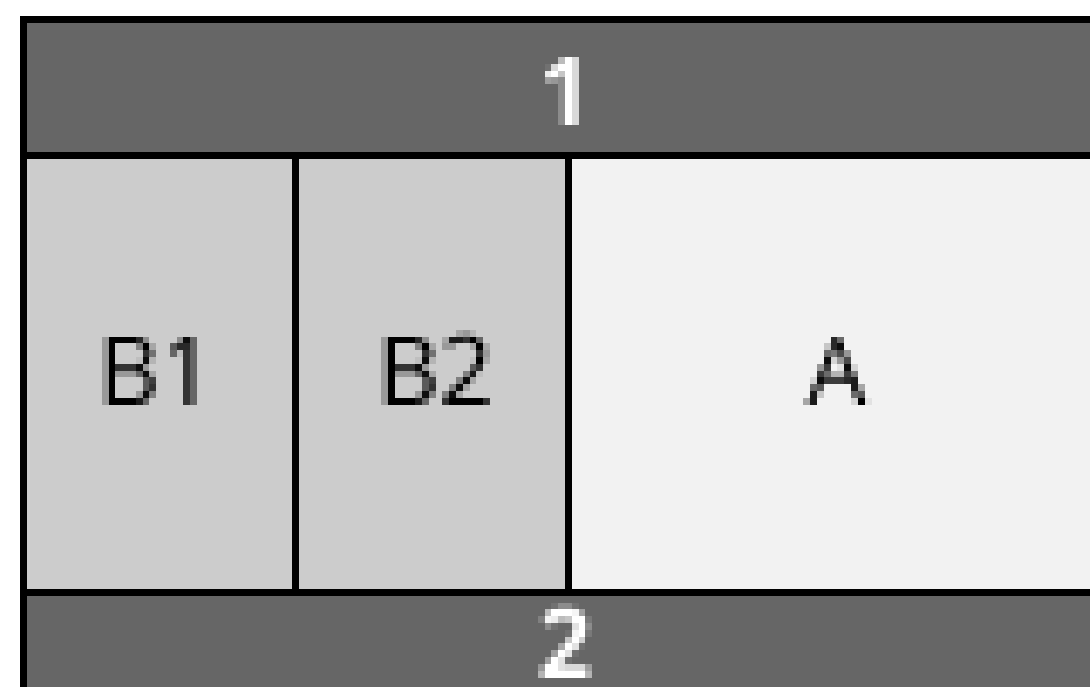
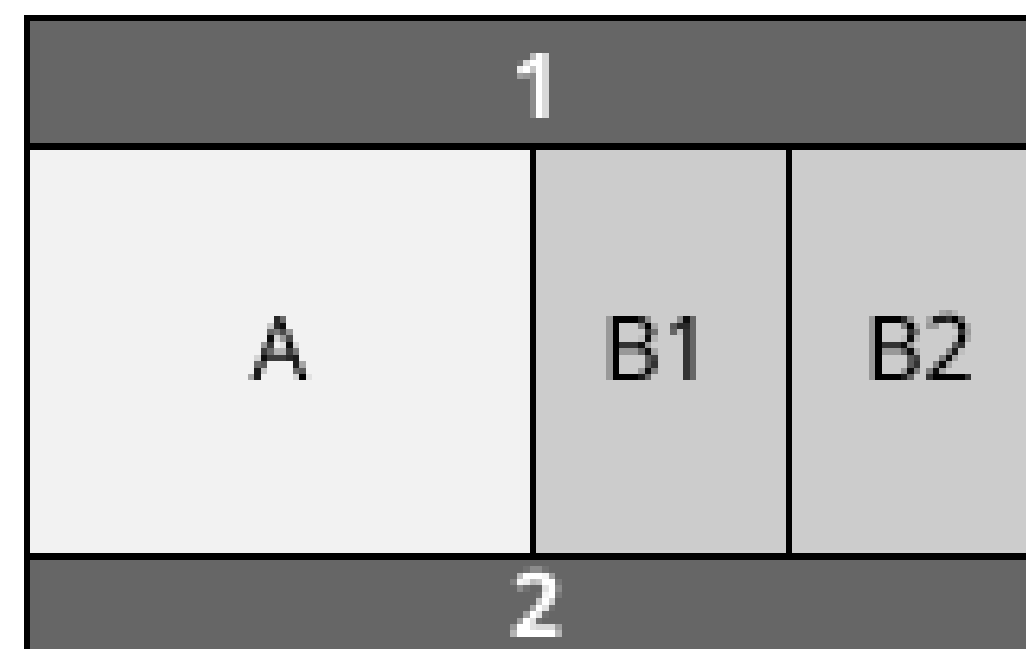
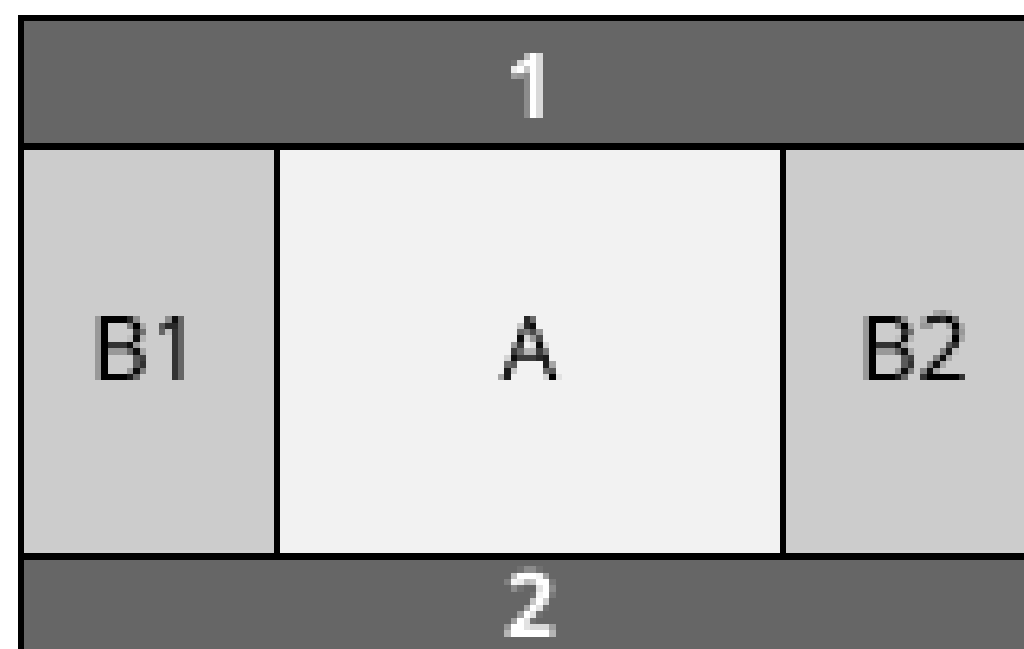
# • COLUMN LAYOUT – 2 column

- **2-column layout.** Often used to target tablets, since they have medium-size screens



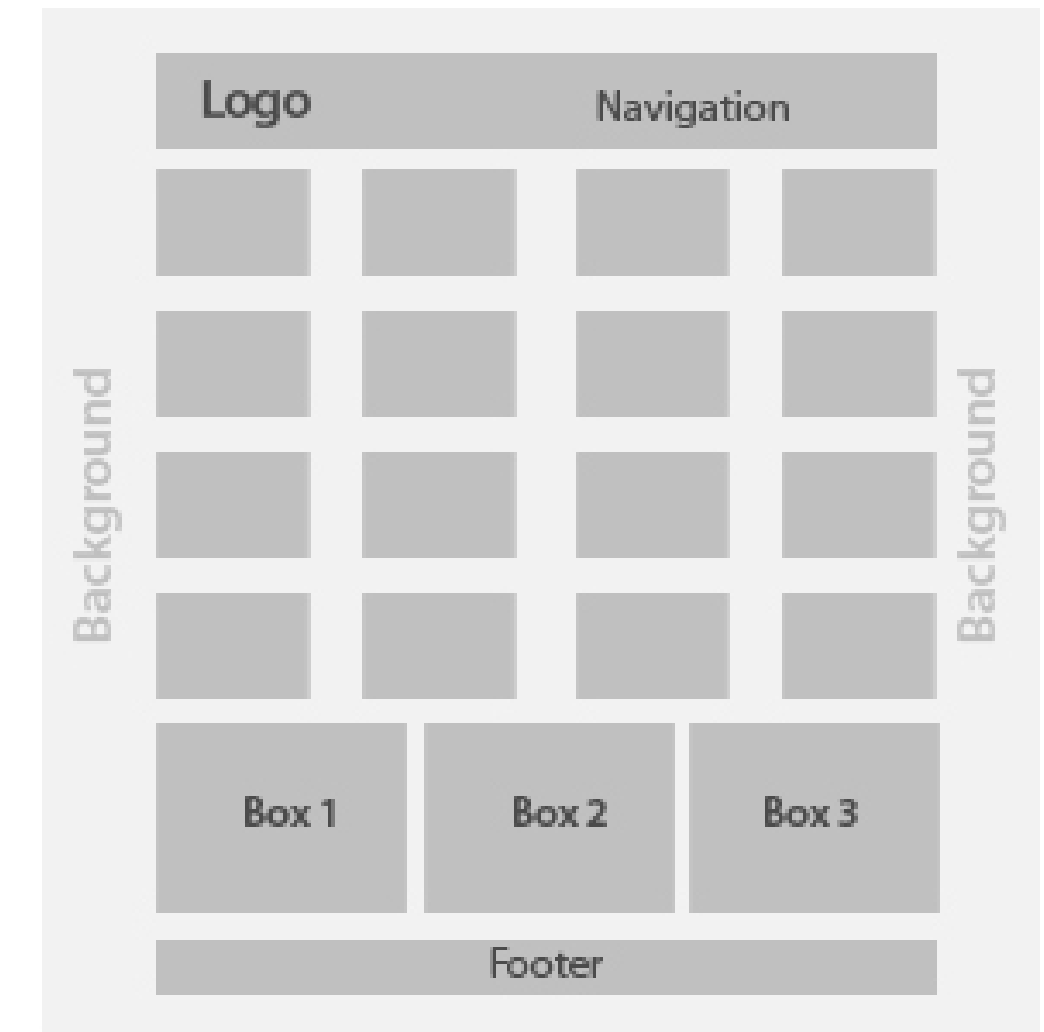
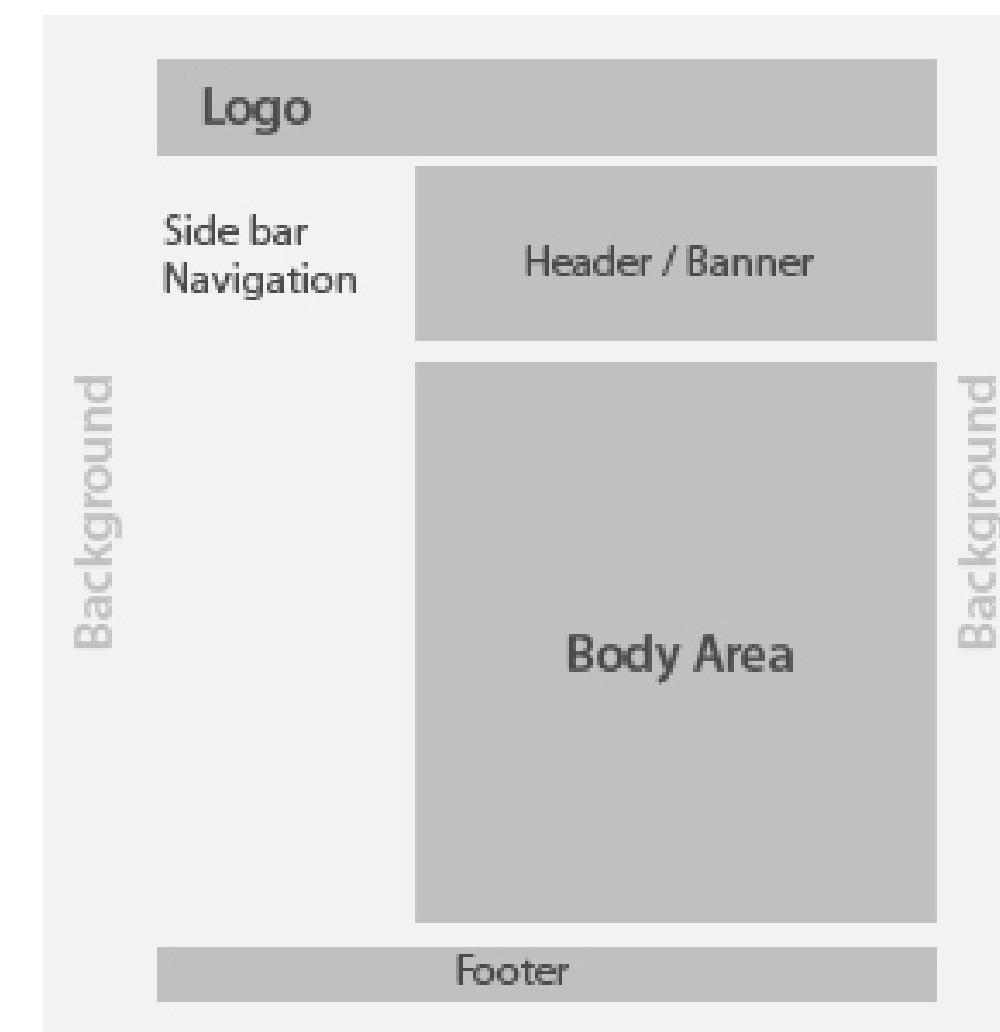
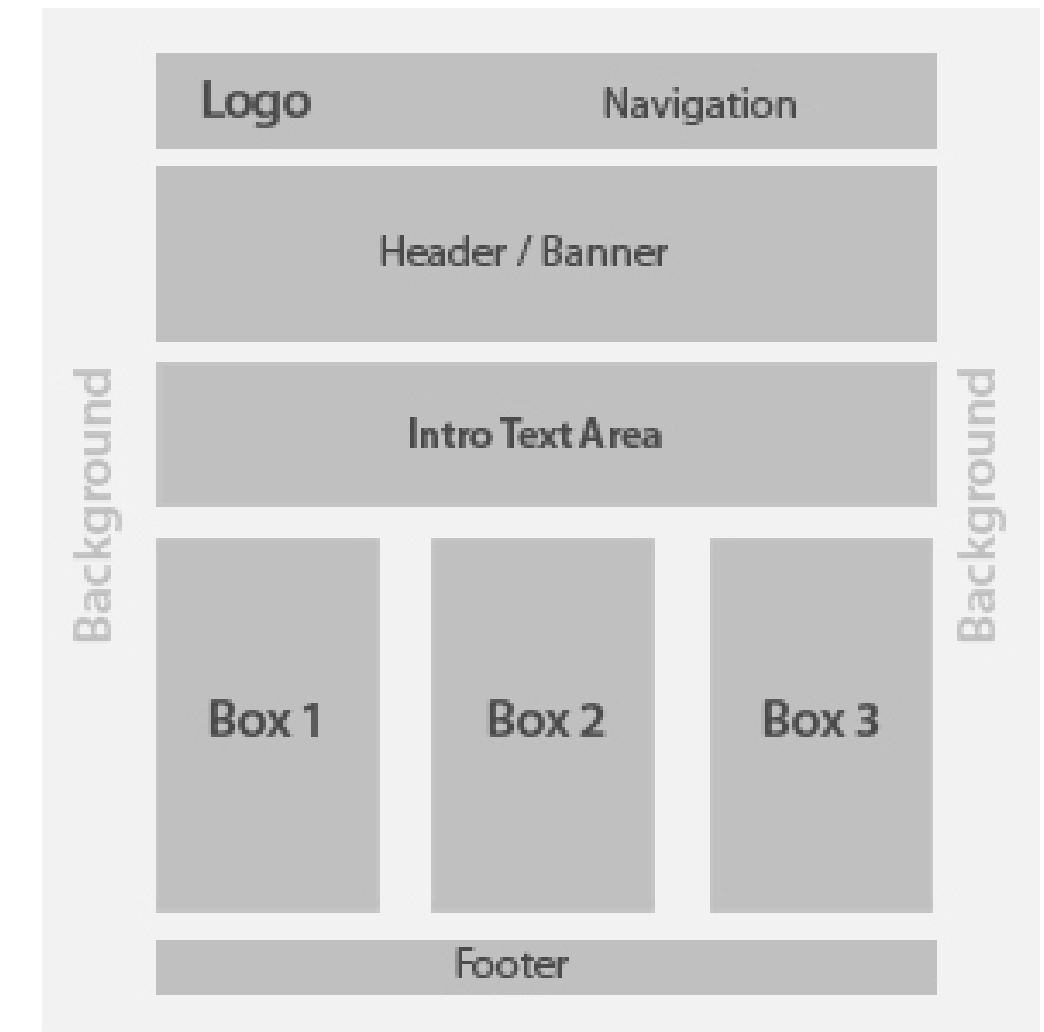
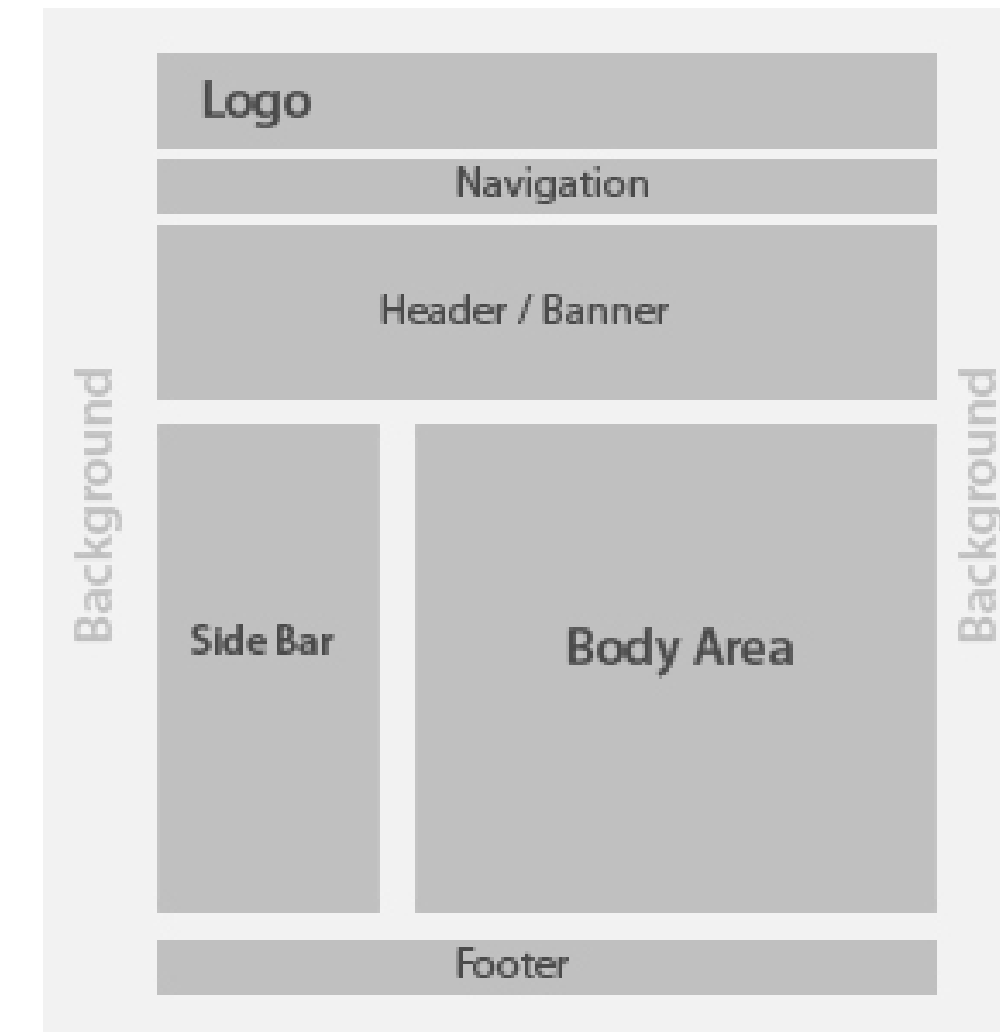
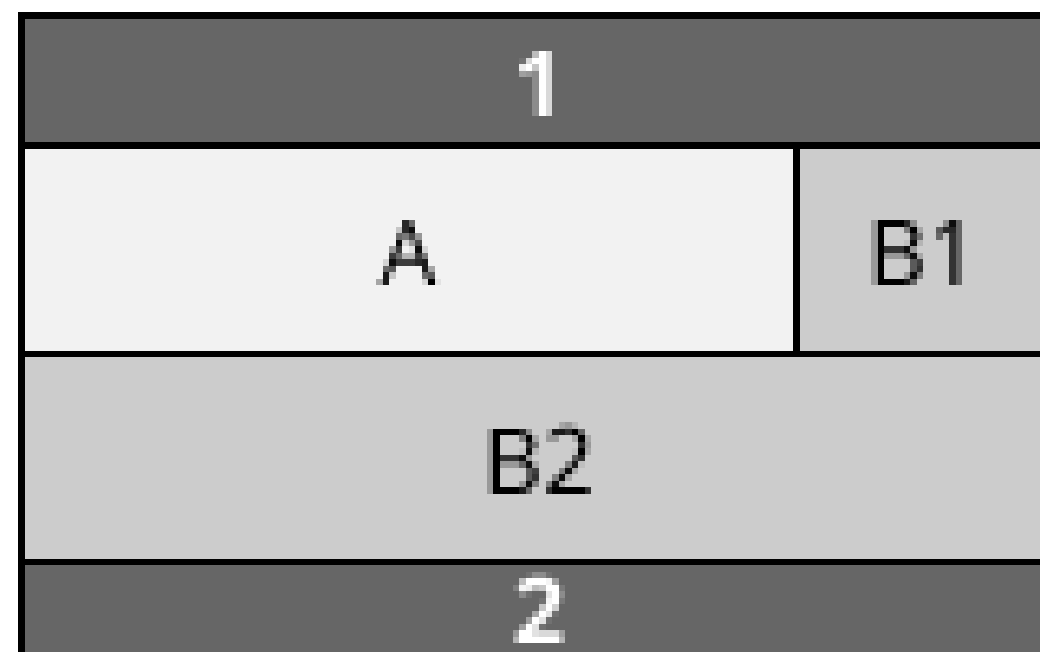
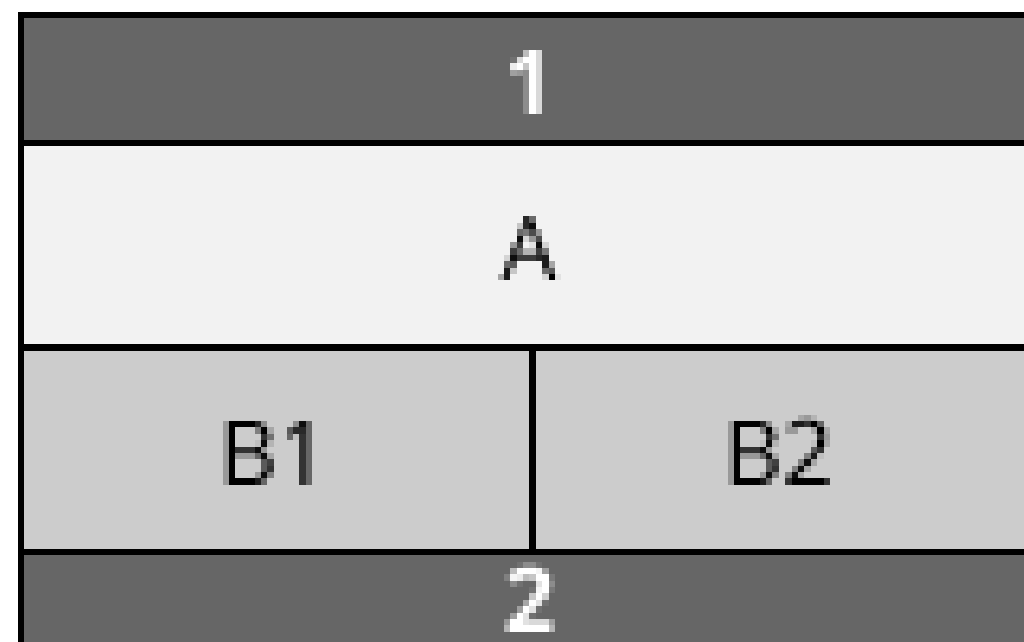
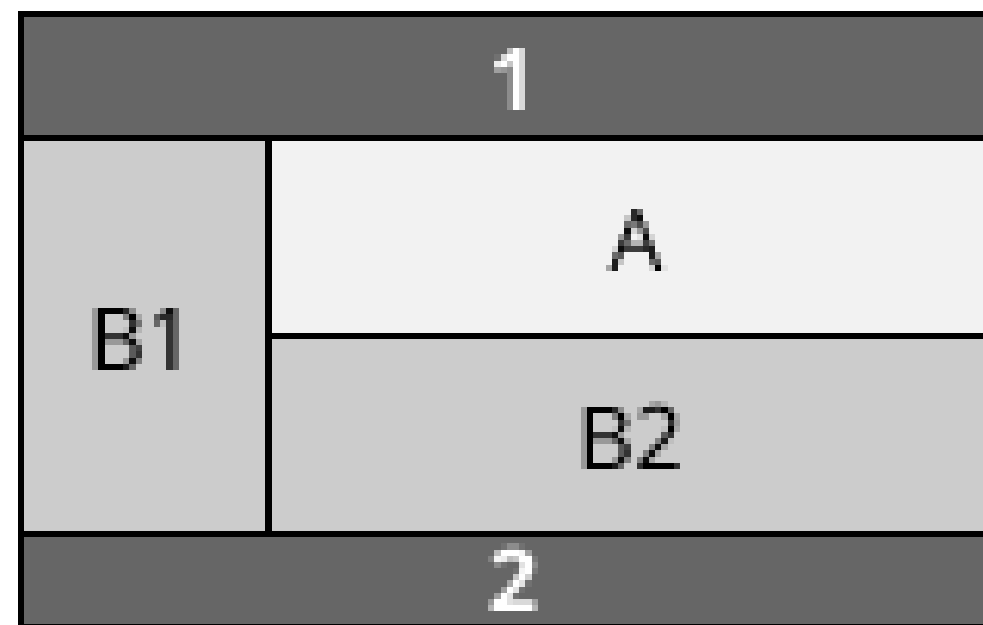
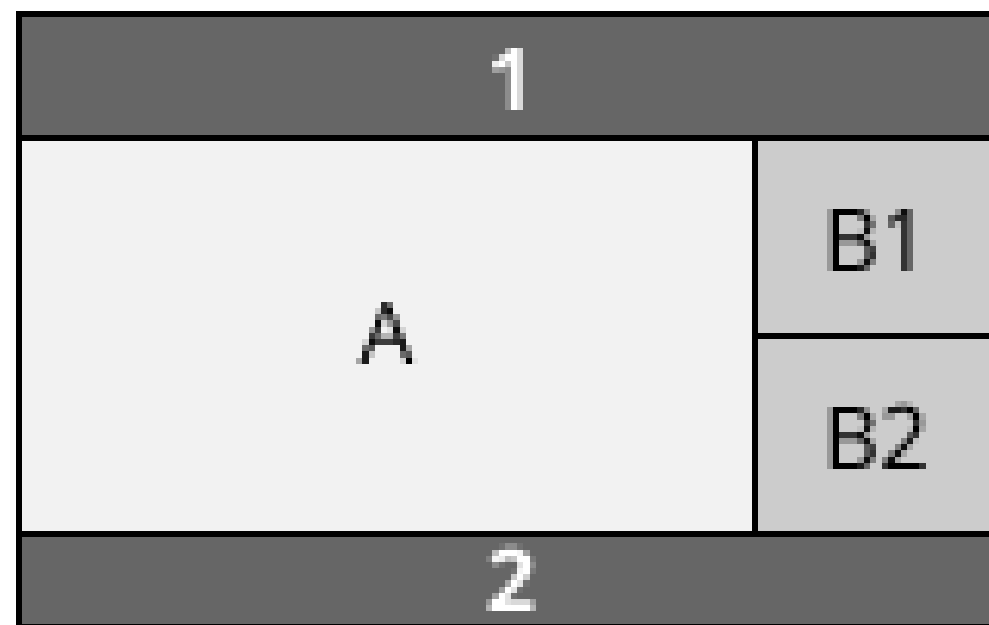
# • COLUMN LAYOUT – 3 column

- **3-column layouts.** Only suitable for desktops with big screens. (Even many desktop-users prefer viewing things in small windows rather than fullscreen.)



# • COLUMN LAYOUT – mix

## • FUN to mix



# •CSS Layout

- CSS page layout techniques allow us to take elements contained in a web page and control where they're positioned relative to the following factors:
  - their default position in normal layout flow,
  - the other elements around them,
  - their parent container,
  - and the main viewport/window

**.The page layout techniques are as follows**

- Normal flow
- The [display](#) property
- Flexbox
- Grid
- Floats
- Positioning
- Table layout
- Multiple-column layout

**Each technique has its uses, advantages, and disadvantages. No technique is designed to be used in isolation. By understanding what each layout method is designed for you'll be in a good position to understand which method is most appropriate for each task**



# •NORMAL FLOW

- Elements on webpages lay themselves out according to normal flow - until we do something to change that.

## Flexbox

a.Flexbox is a one-dimensional layout method for laying out items in rows or columns

## Grids

a.CSS Grid Layout is a two-dimensional layout system for the web. It lets you lay content out in rows and columns, and has many features that make building complex layouts straightforward

## Floats

a.Originally for floating images inside blocks of text, the [float](#) property became one of the most commonly used tools for creating multiple column layouts on webpages. With the advent of Flexbox and Grid it has now returned to its original purpose, as this article explains.

## Positioning

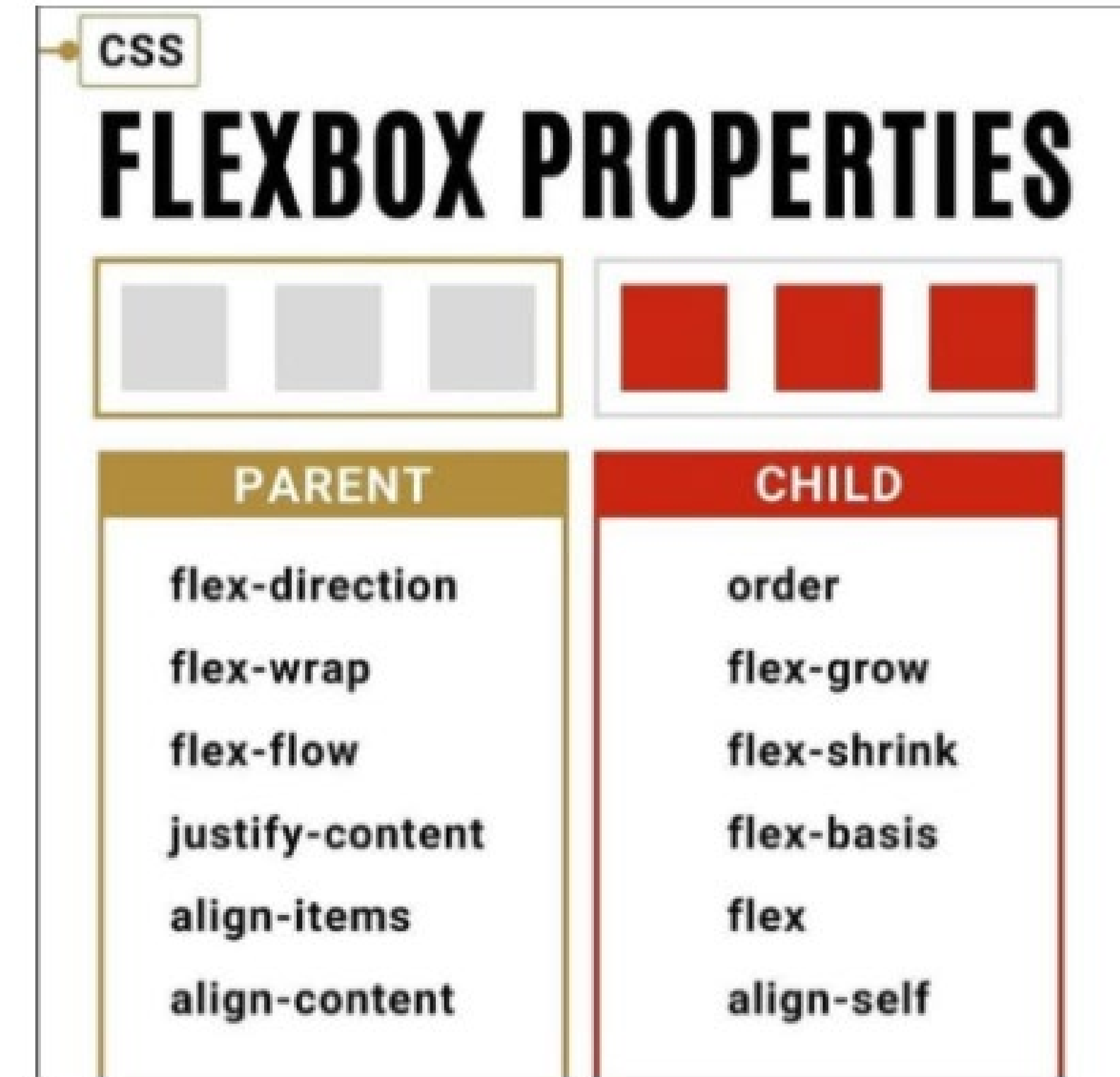
a.Positioning allows you to take elements out of the normal document layout flow and make them behave differently, for example, by sitting on top of one another, or by always remaining in the same place inside the browser viewport.

## Multiple-column

a.The multiple-column layout specification gives you a method of laying content out in columns, as you might see in a newspaper

# FLEX BOX

- The CSS Flexbox is a one-dimensional layout. It is useful in allocating and aligning the space among items in a grid container.
- It works with various kinds of display devices and screen sizes.
- Flexbox layout makes it easier to design and build responsive web pages without using many float and position properties in the CSS code.



# •GRID

A grid is a collection of horizontal and vertical lines creating a pattern against which we can line up our design elements.

- They help us to create layouts in which our elements won't jump around or change width as we move from page to page, providing greater consistency on our websites.
- A grid will typically have columns, rows, and then gaps between each row and column.
- The gaps are commonly referred to as gutters.

```
css grid

.grid-parent {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 100px 100px;
  grid-template-areas:
    "header header header"
    "aside content content";
  grid-auto-columns: 110px;
  grid-auto-rows: 120px;
  grid-auto-flow: row;
  grid-column-gap: 10px;
  grid-row-gap: 10px;
  grid-gap: 10px;
  align-items: center;
  align-content: flex-start;
  justify-items: center;
  justify-content: flex-start;
}

.grid-parent .grid-child {
  grid-area: header;
  grid-column: 2 / 3;
  grid-column-start: 2;
  grid-column-end: 3;
  grid-row: 1 / 2;
  grid-row-start: 1;
  grid-row-end: 2;
  align-self: center;
  justify-content: flex-end;
  order: 2;
}
```

@code4mk

# When to use

Use **Flexbox** when you need to create simple, one-dimensional layouts where items flow in a single direction, such as navigation menus or aligning items in a row or column.


Use **CSS Grid** when you need to create more complex, two-dimensional layouts with precise control over both rows and columns, like grid-based page layouts, card-based designs, and complex grid structures.

In many cases, both Flexbox and Grid can complement each other, with Flexbox used within Grid items to handle their internal layout. It's essential to choose the layout technique that best suits your specific design requirements.



# CSS Flexbox vs CSS Grid

## Comparison Chart

CSS Flexbox	CSS Grid
CSS Flexbox is a one-dimensional layout model.	CSS Grid is a two-dimensional model.
A Flexbox container can either facilitate laying out things in a row, or lay them out in a column.	Grid can facilitate laying out items across and down at once.
Flexbox cannot intentionally overlap elements or items in a layout.	CSS Grid helps you create layouts with overlapping elements.
Flexbox is basically content based and it listens to the content and adjusts to it.	Grid operates more on the layout level and it is container based.
Flexbox can be used for scaling, one-sided aligning, and organizing elements within a container.	Grid is useful when you want to define a large-scale layout with more complex and subtle designs.
	



# CLASS ACTIVITIES

# FLEX exercise 01

- Download flex01.html
- Add the following code into .container {}

CODE		Observe
1	display: flex;	
2	flex-direction: row;  Then Change the value row to: <ul style="list-style-type: none"> <li>• Row-reverse</li> <li>• Column</li> <li>• Colum-reverse</li> </ul>	
3	flex-wrap: nowrap;  Then change the value “nowrap” to “wrap” and resize your browser	

# FLEX exercise 02

- Download flex02.html and layout.css

<https://developer.mozilla.org/en-US/docs/Web/CSS/flex>

CODE		Observe
1	Create the link to layout.css in flex02.html	
2	Insert <i>display:flex;</i> in the Parent flex	
3	Insert the following values in each of the Child <ul style="list-style-type: none"> <li>• flex:4;</li> <li>• flex:1;</li> <li>• flex:1;</li> </ul> (play around with the numbers to get what layout you want)	
4	Insert the following values in each of the Child <ul style="list-style-type: none"> <li>• order:1;</li> <li>• order:2;</li> <li>• order:3;</li> </ul> (play around with the numbers to get what layout you want)	





# CLASS ACTIVITIES

# GRID 01 Exercise

Download grid.html  
and mygrid.css

CODE		Observe
1	Create the link to mygrid.css in grid.html	
2	Insert the following in the Parent <pre>display: grid; grid-template-columns: 1fr 1fr 1fr 1fr; gap: 20px;</pre> * Then tick the Grid display setting in the inspector view	
3	<pre>header{   grid-column: 1/span 4;   grid-row: 1; }</pre>	
4	<pre>nav{   grid-column: 1;   grid-row: 2; } article{   grid-column: 2/span 2; } aside{   grid-column: 4; } footer{   grid-column: span 4;   grid-row: 3; }</pre>	

# GRID 01 Exercise

Download grid.html  
and mygrid.css

CODE		Observe
1	Create the link to mygrid.css in grid.html	
2	Insert the following in the Parent <pre>display: grid; gap: 20px; grid-template-areas: "head head head head" "navi navi navi navi" "content content content side" "foot foot foot foot" ; grid-template-columns: 1fr 1fr 1fr 1fr;</pre>	
3	<pre>header{   grid-area: head; }  article{   grid-area: content; } nav{   grid-area: navi; } aside{   grid-area: side ; } footer{   grid-area: foot; }</pre>	