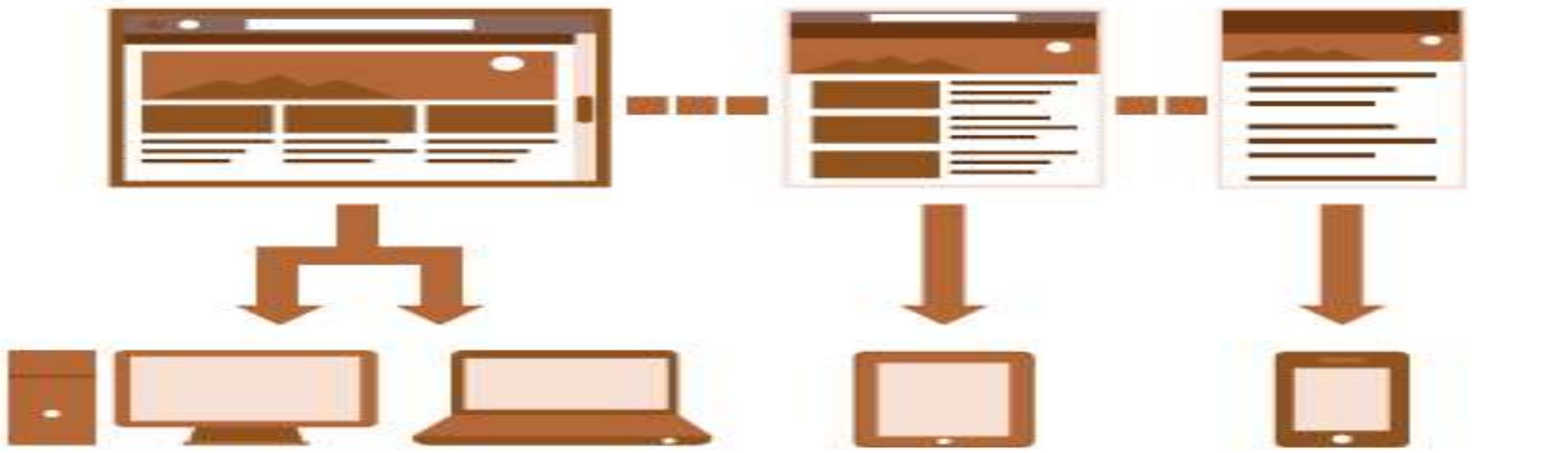
Responsive Web Design

ENG/RYHAB FAROUQ

Meaning of RWD

Responsive design teams create a single site to support many devices, but need to consider content, design and performance across devices to ensure usability.



RWD

Responsive Web Design was first introduced by Ethan Marcotte in 2009.

Welcome to Ethan Marcotte’s website — Ethan Marcotte

Think responsively.. Think Mobile First

Viewport

Viewport is defined area to display website independent on device screen.

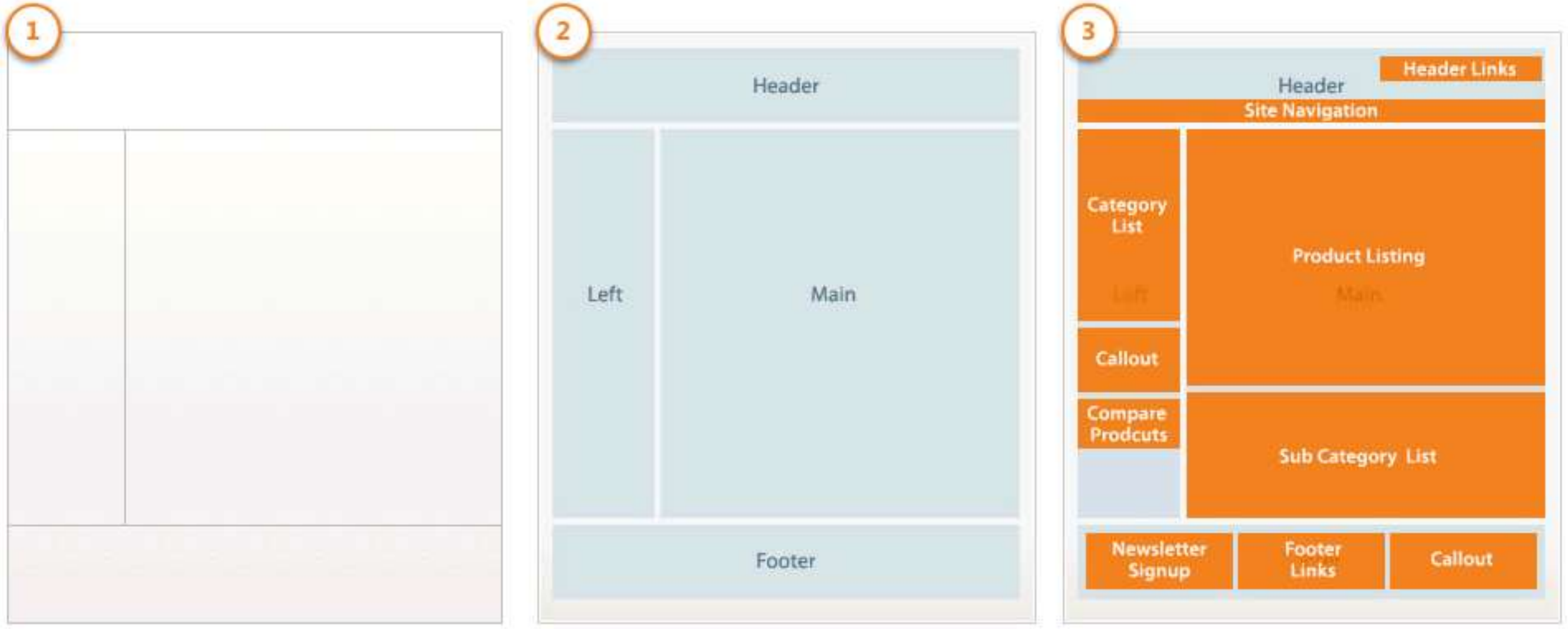
Identify and control viewport and its initial scale factor to override useragent.

```
<meta name="viewport" content=""/>
```

Terminologies

- **Responsive design** is where website isn’t fixed with single size, It responds to users’ device automatically
- **Adaptive design** is where created website redesign itself as per the device size
- **Negative space** is empty space between elements to it more readable & standout
  - padding or margins are great strategy to create it

Web Layout


1
2
Header
Left
Main
Footer
3
Header
Header Links
Site Navigation
Category List
Left
Product Listing
Main
Callout
Compare Prodcuts
Sub Category List
Newsletter Signup
Footer Links
Callout

RWD Fundamentals

- Flexbox
- Grid system
- Media queries
- Responsive frameworks ‘bootstrap’

Flexbox

Flexbox

One dimensional layout model .

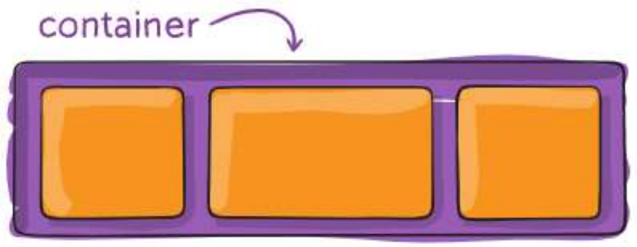
- Flexible and efficient layouts
- Distribute space among items
- Control their alignment

Why Flexbox ?

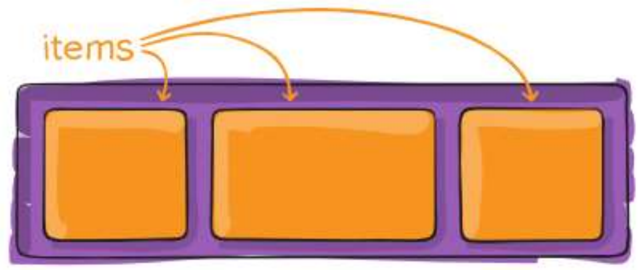
- A lot of flexibility
- Arrange items
- Spacing
- Alignment
- Order of items
- Bootstrap is built on top of the flex layout

Terminology

- Flex container

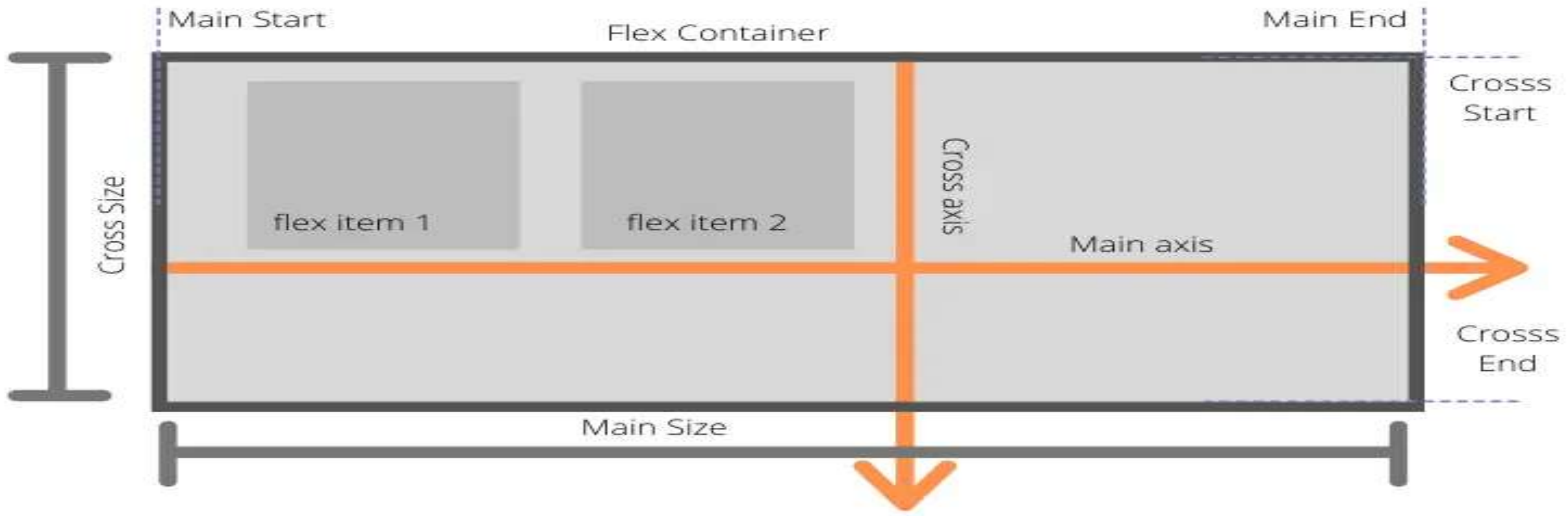


- Flex items



Flexbox Axes

- Main axis
- Cross axis



Flex Container Properties

- Display
- Flex-direction
- Flex-wrap
- Flex-flow
- Justify-content
- Align-items
- Align-content

Display

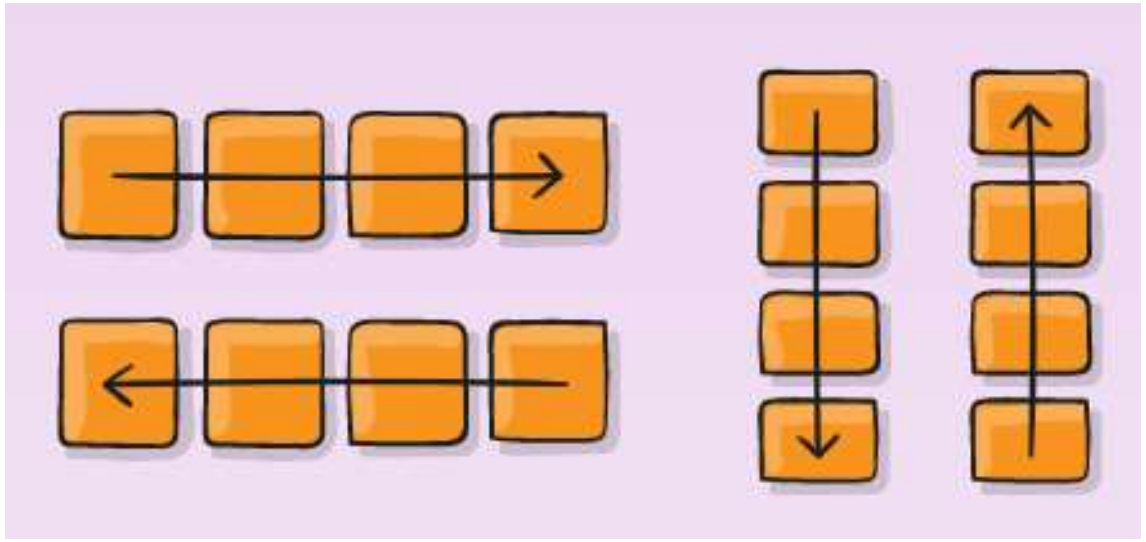
Create either a block level or inline level flex container.

- Flex
- Flex-inline

Flex-direction

Sets the direction of the main axis.

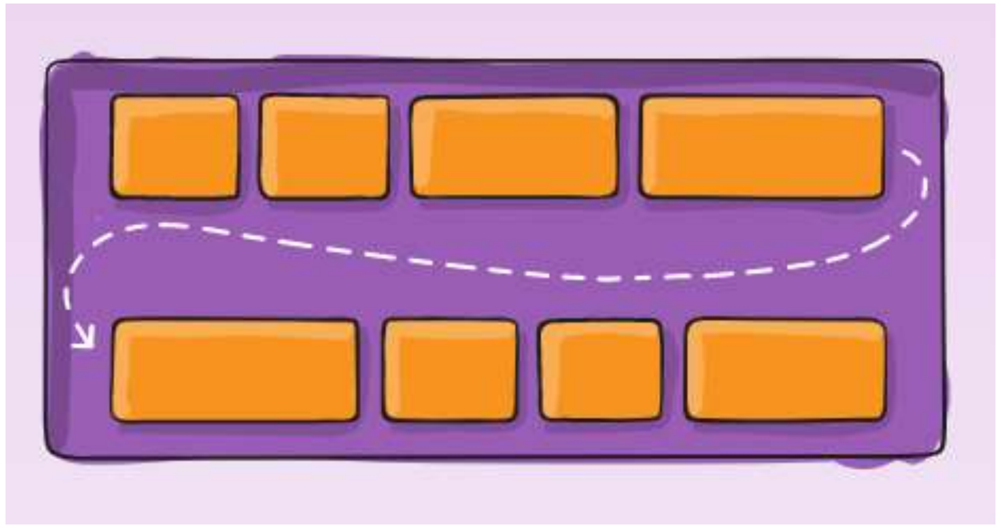
- Row ➔ Default value
- Row-reverse
- Column
- Column-reverse



Flex-wrap

Control the wrapping of flex items within the container.

- Nowrap ➔ Default value
- Wrap
- Wrap-reverse ➔ push the last item above instead of below



Flex-flow

Shorthand for flex-direction and flex-wrap.

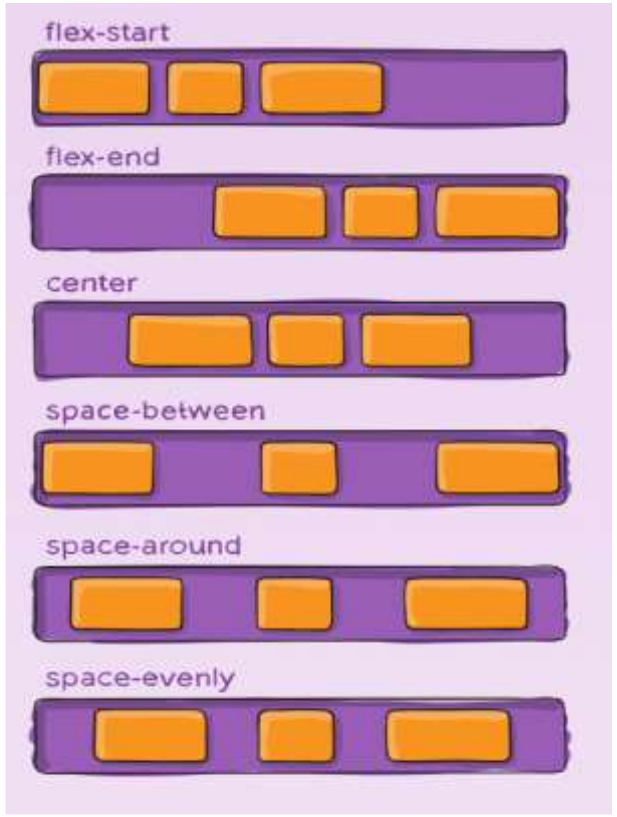
- Flex-flow **:** <flex-direction> <flex-wrap> ➔ the Default is row nowrap

Justify-content

Align items and distribute any extra spacing in the parent container.

The alignment is always along the main axis.

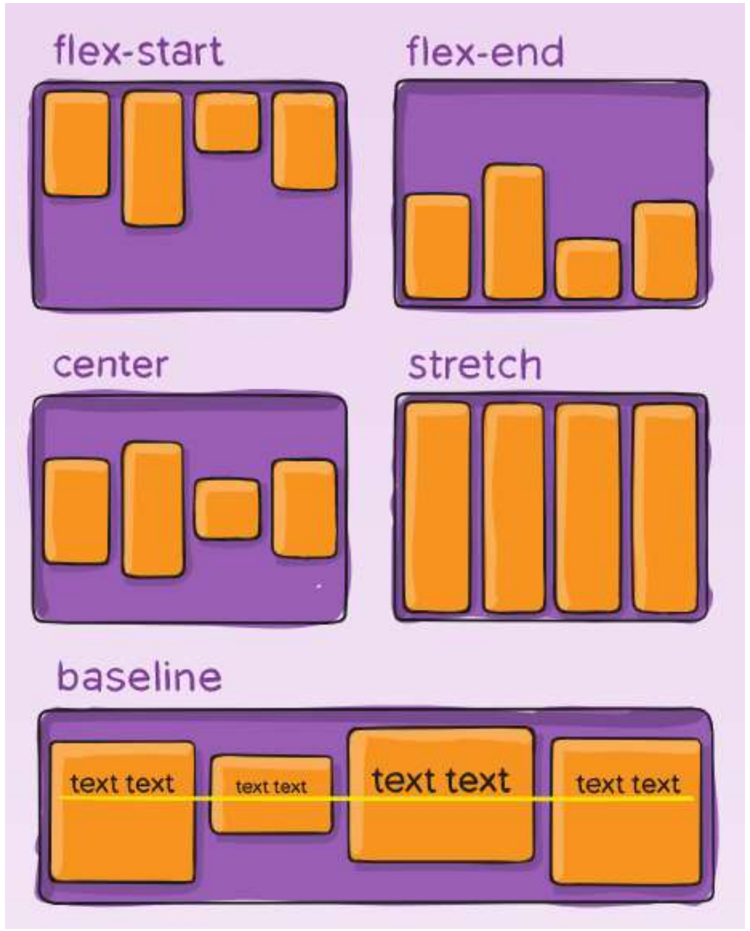
- Flex-start ➔ Default value
- Flex-end
- Center
- Space-between
- Space-around
- Space-evenly



Align-Items

Align items along the cross axis.

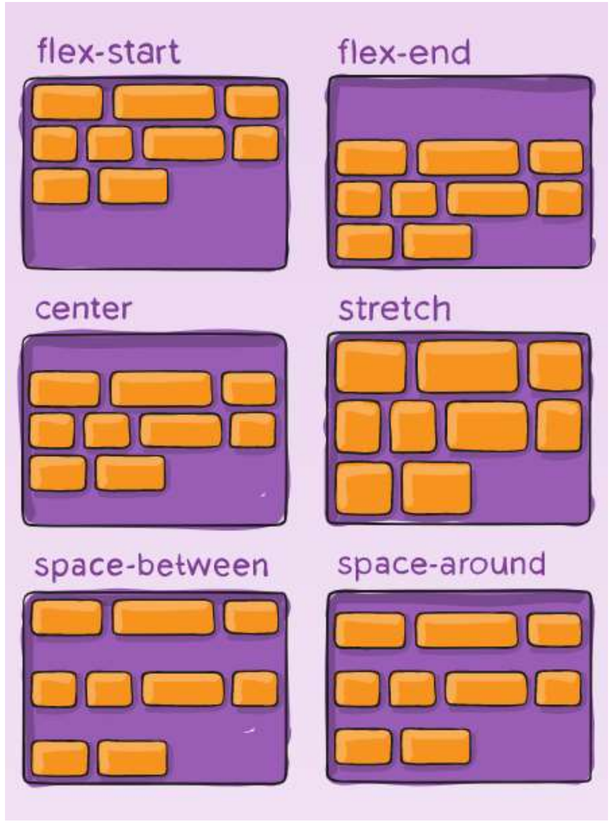
- Flex-start
- Flex-end
- Center
- Baseline
- Stretch → Default value



Align-Content

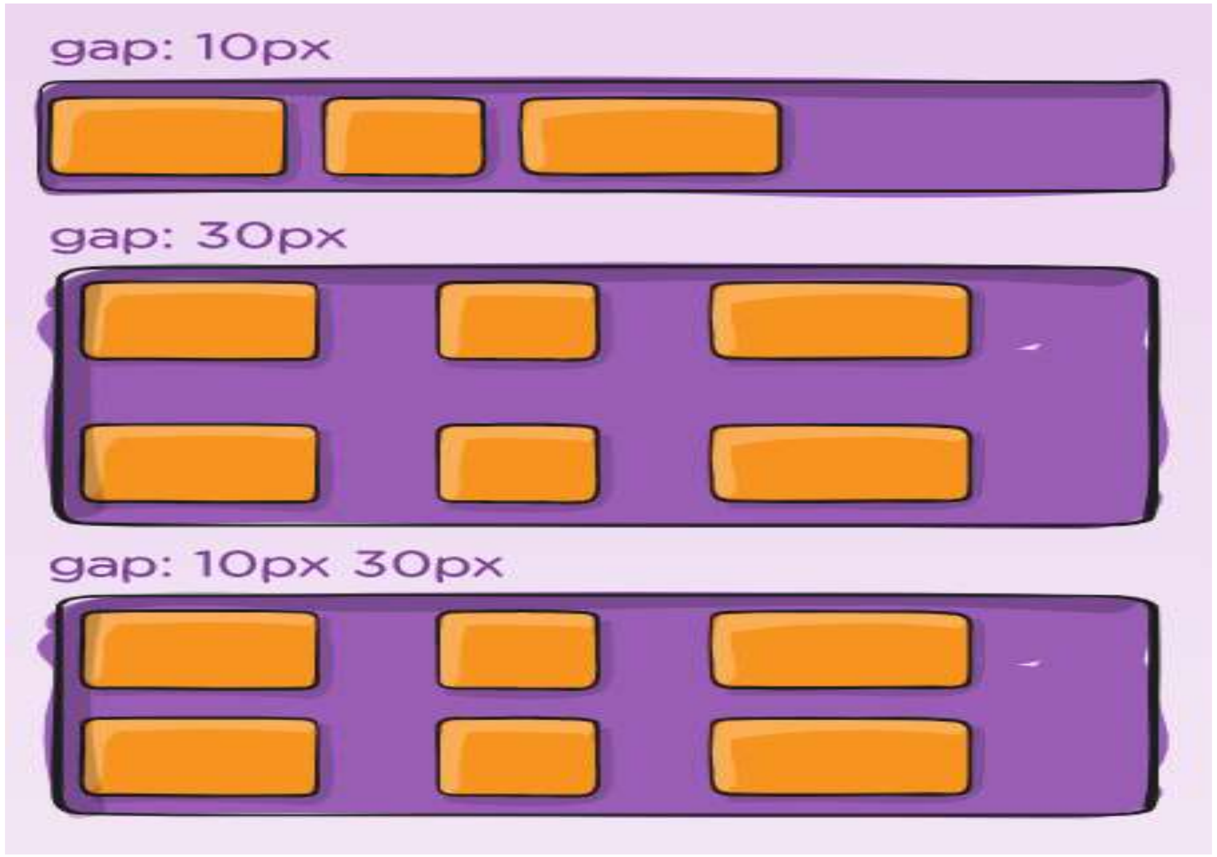
Aligns lines of content along the cross axis and distribute any extra spacing in the parent container .

- Flex-start
- Flex-end
- Center
- Space-between
- Space-around
- Stretch ➔ Default value



Gap

- gap : row column



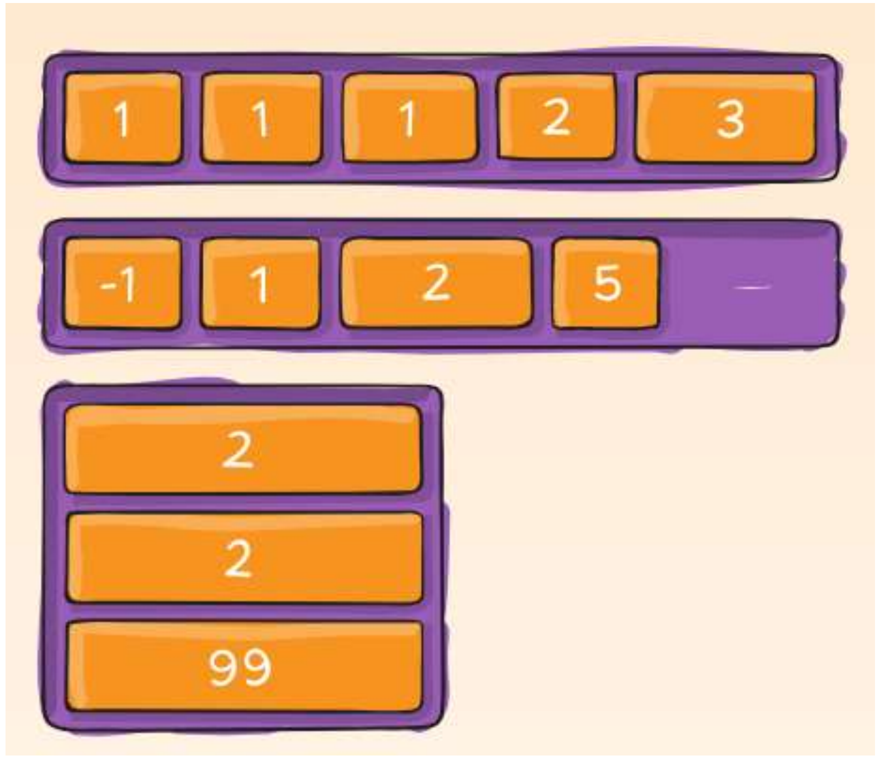
Flex Items Properties

- Order
- Flex-grow
- Flex-shrink
- Flex-basis
- Flex
- Align-self

Order

Control the order of items in the flex container .

Zero is the default value .

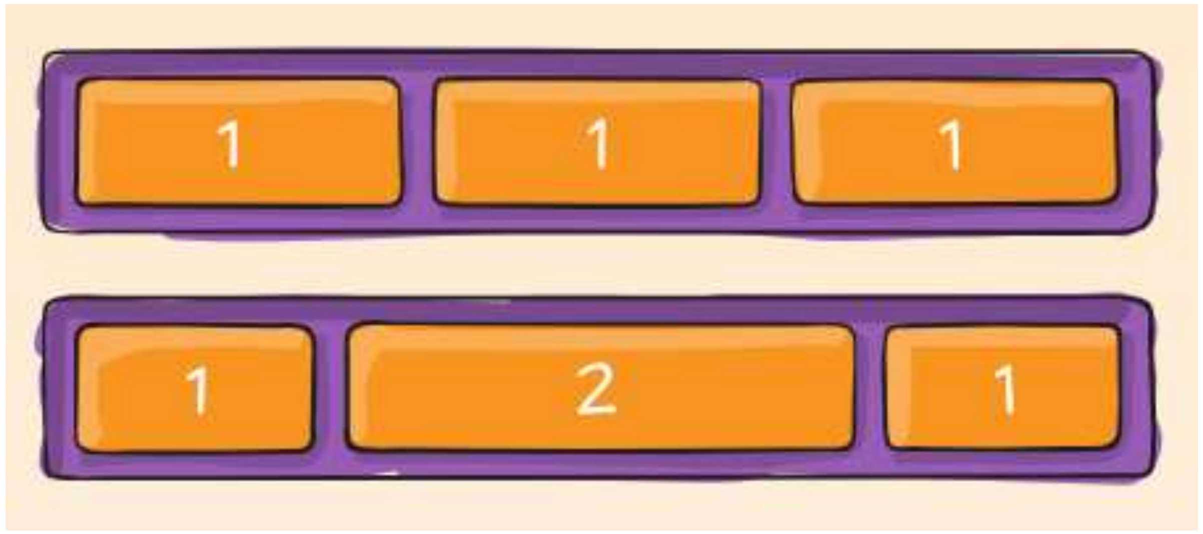


Flex-Grow

Dictates what amount of the available space inside the flex container the items should take up.

Relative to other items in the container.

Default value is zero .



Flex-Shrink

Dictates the shrink factor of the flex items when the default size of flex items is larger than the flex container .

Relative to other items in the container in case of increasing shrink .

Default value is 1 .

Flex-Basis

Set the initial size of flex items.

It accepts pixels , percentages or relative units .

Flex

Shorthand for flex-grow , flex-shrink and flex-basis .

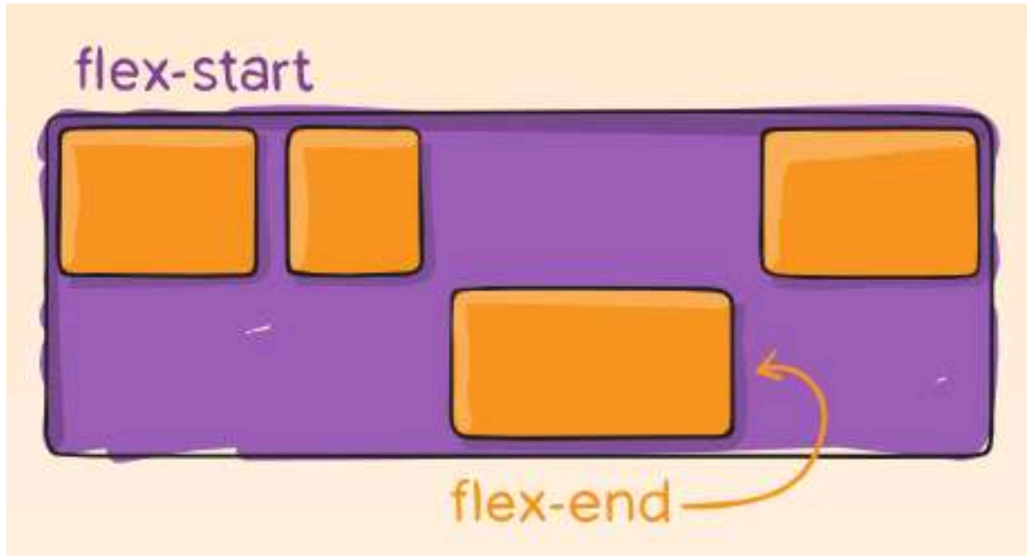
Flex : <flex-grow> <flex-shrink> <flex-basis>

Align-Self

Align items individually .

Overrides the align-items value of the flex container

- Auto
- Flex-start
- Flex-end
- Center
- Stretch



Grid System

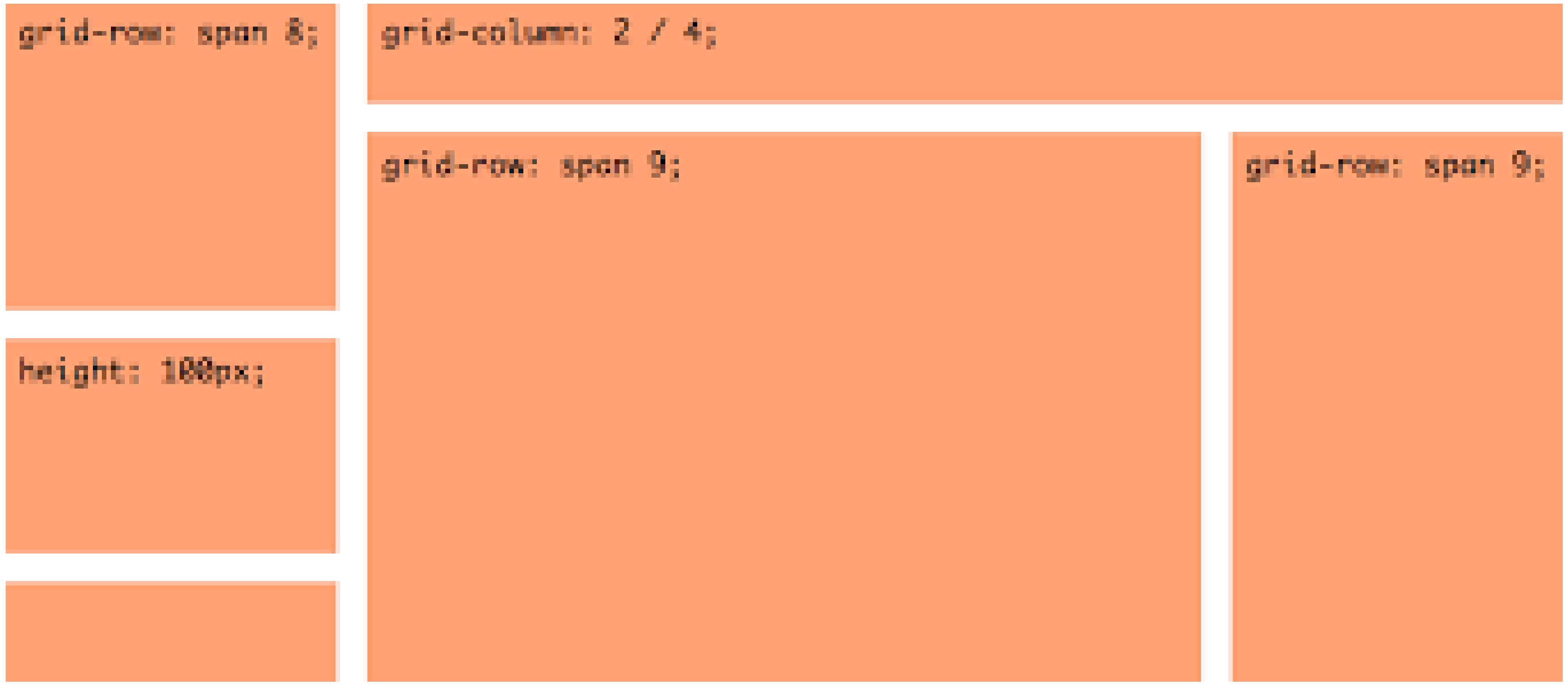
Grid System CSS

Two dimensional layout model .

CSS Grid will allow you to manage layout according to both *columns* and *rows*.

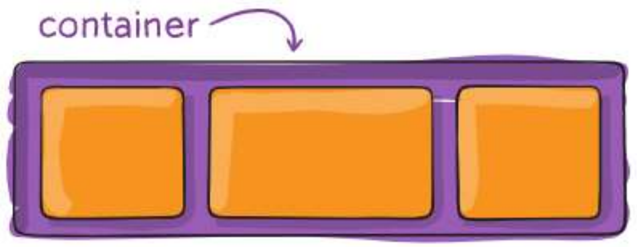
Flexbox, by comparison, is really just meant for columns (1-D, if you will). Using CSS Grid, designers will be able to achieve layouts that were downright difficult before. In fact, some of these things were on our wish lists since the days of table-based layouts.

2-D Layout

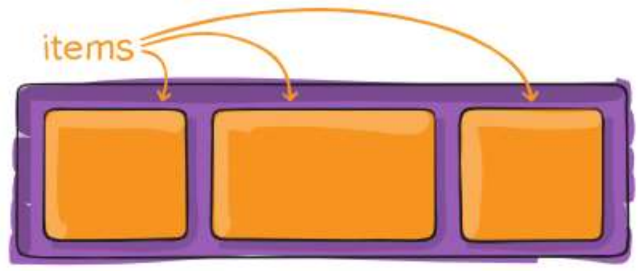

grid-row: span 8;
grid-column: 2 / 4;
grid-row: span 9;
grid-row: span 9;
height: 100px;

Terminology

- container



- items



Two Ways to implement Grid in CSS

- Way_one
  - Grid-template-columns
  - Grid-template-rows
  - Grid-column : start/end
  - Grid-row : start/end

Two Ways to implement Grid in CSS

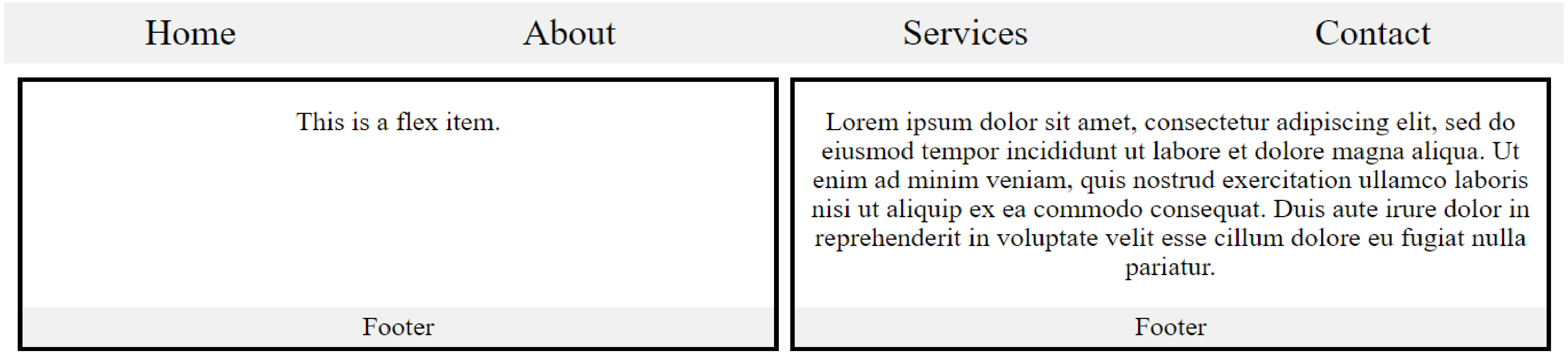
- Way_two
  - Grid-template-areas
  - Grid-area

Media Queries

Media Queries

- Tell the browser how to style an element at particular viewport dimensions
- Types :
  - all
  - Screen
  - Print
  - Speech

Task_One



Home About Services Contact

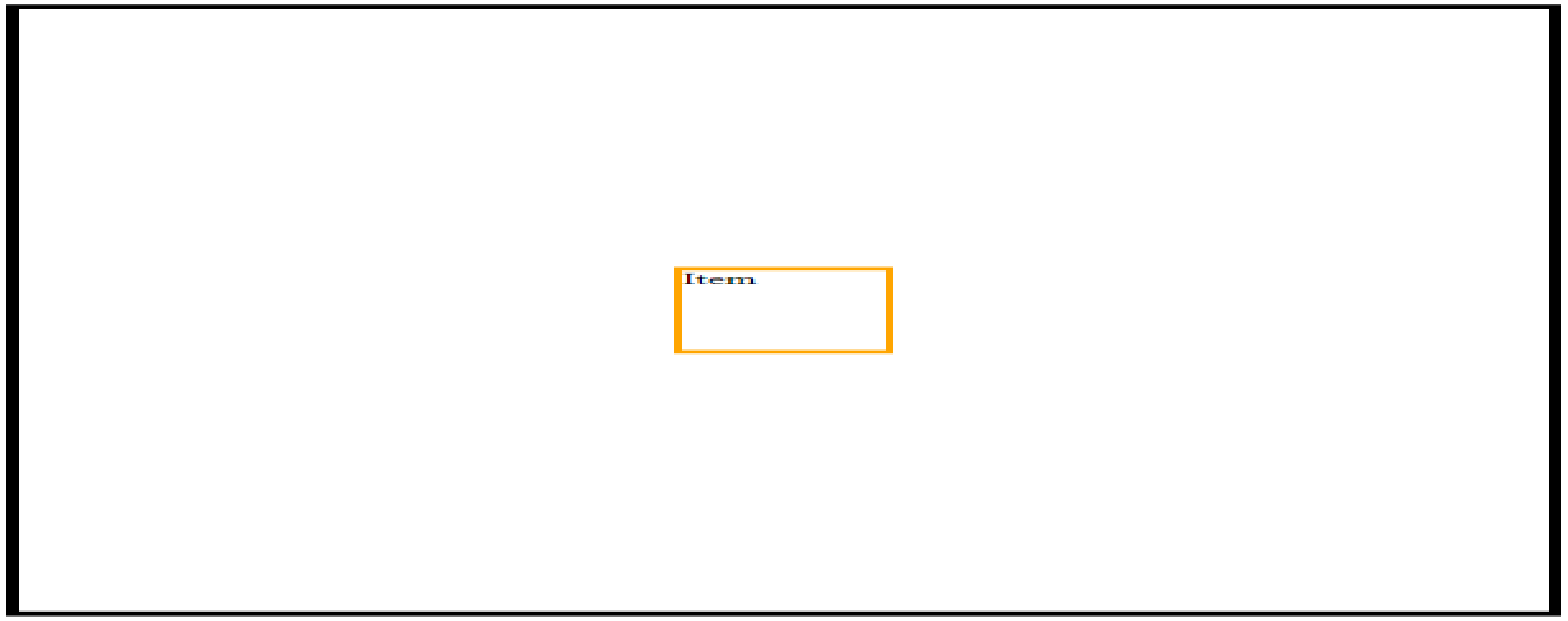
This is a flex item.

Footer

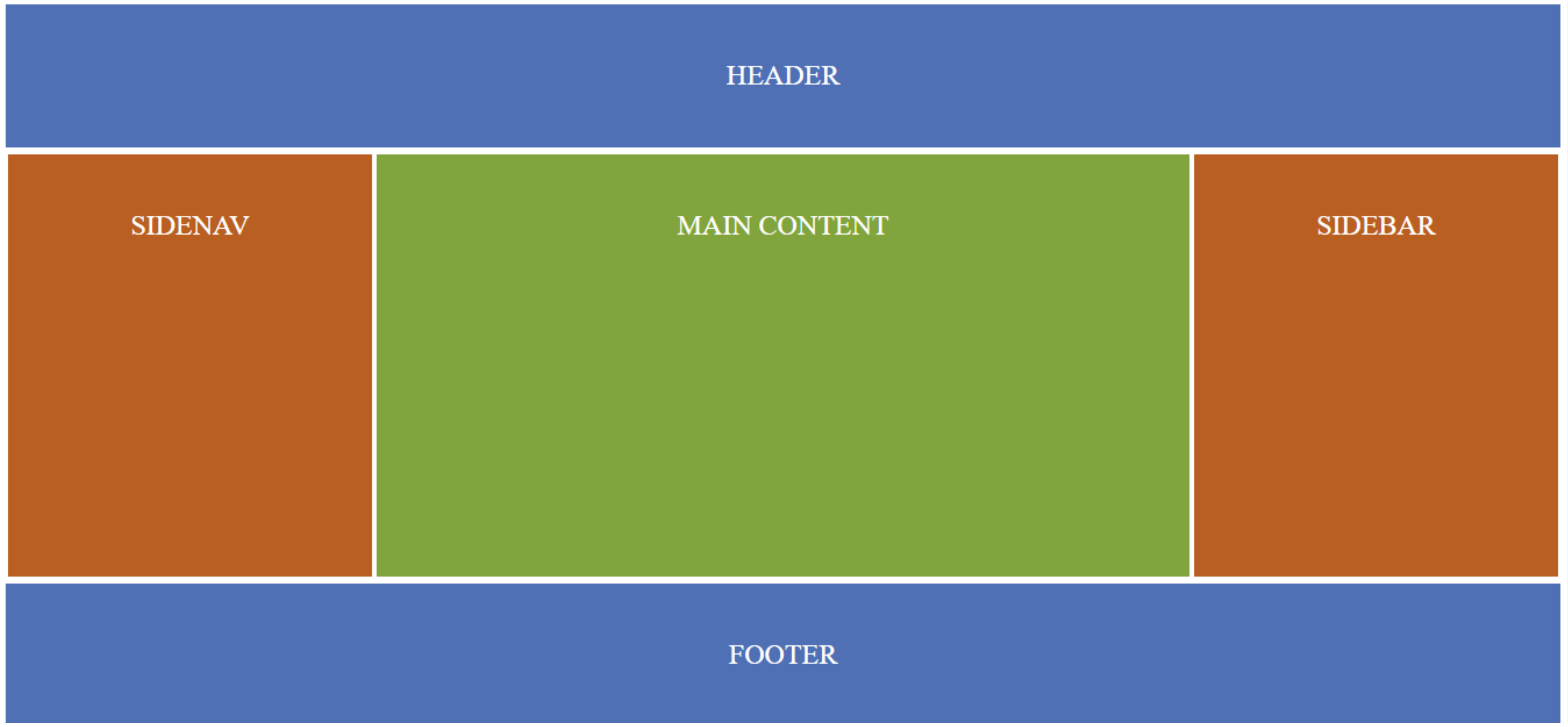
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Footer

Task_Two



Task_Three



Task_Four