CSS

CSS – Flex Items

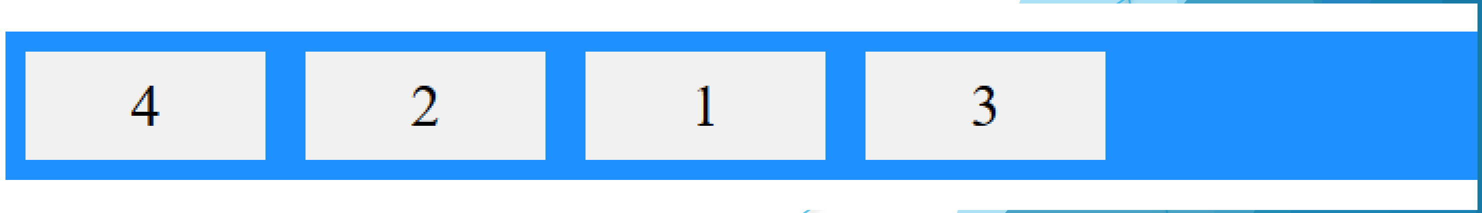
The flex items can have the following properties:

- **order -** Specifies the display order of the flex items inside the flex container
- **flex-grow -** Specifies how much a flex item will grow relative to the rest of the flex items
- **flex-shrink -** Specifies how much a flex item will shrink relative to the rest of the flex items
- **align-self -** Specifies the alignment for the flex item inside the flex container

order - the display order of the flex items inside the flex container.

```
<style>

.container {

  display: flex;

  background-color: dodgerblue; }

.container div {

  background-color: #f1f1f1;

  color:#000;

  width: 100px;

  margin: 10px;

  padding: 10px;

  text-align: center;

  font-size: 30px; }

</style>
```

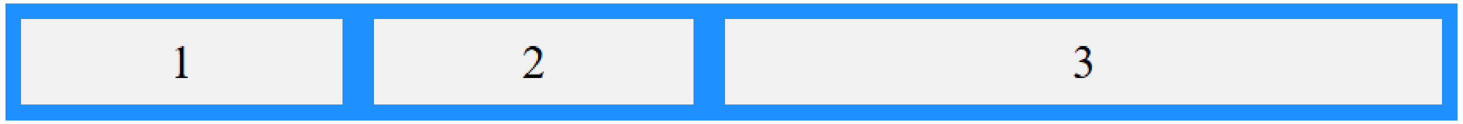
```
<div class="container">

  <div style="order: 3">1</div>

  <div style="order: 2">2</div>

  <div style="order: 4">3</div>

  <div style="order: 1">4</div>

</div>
```



flex-grow - how much a flex item will grow relative to the rest of the flex items.

```
<style>

.container {

  display: flex;

  background-color: dodgerblue; }

.container div {

  background-color: #f1f1f1;

  color:#000;

  width: 100px;

  margin: 10px;

  padding: 10px;

  text-align: center;

  font-size: 30px; }

</style>
```

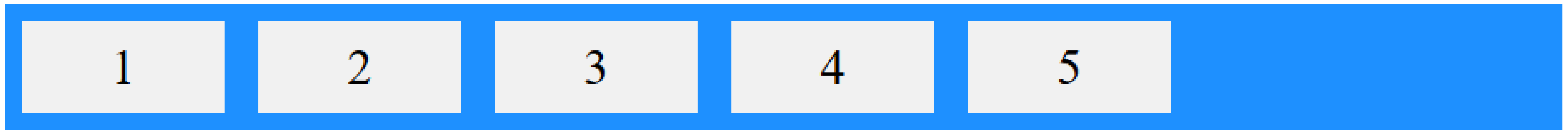
```
<div class="container">

  <div >1</div>

  <div style="flex-grow: 1">2</div>

  <div style="flex-grow: 2">3</div>

</div>
```



flex-shrink - how much a flex item will shrink relative to the rest of the flex items.

```
<style>

.container {

  display: flex;

  background-color: dodgerblue; }

.container div {

  background-color: #f1f1f1;

  color:#000;

  width: 100px;

  margin: 10px;

  padding: 10px;

  text-align: center;

  font-size: 30px; }

</style>
```

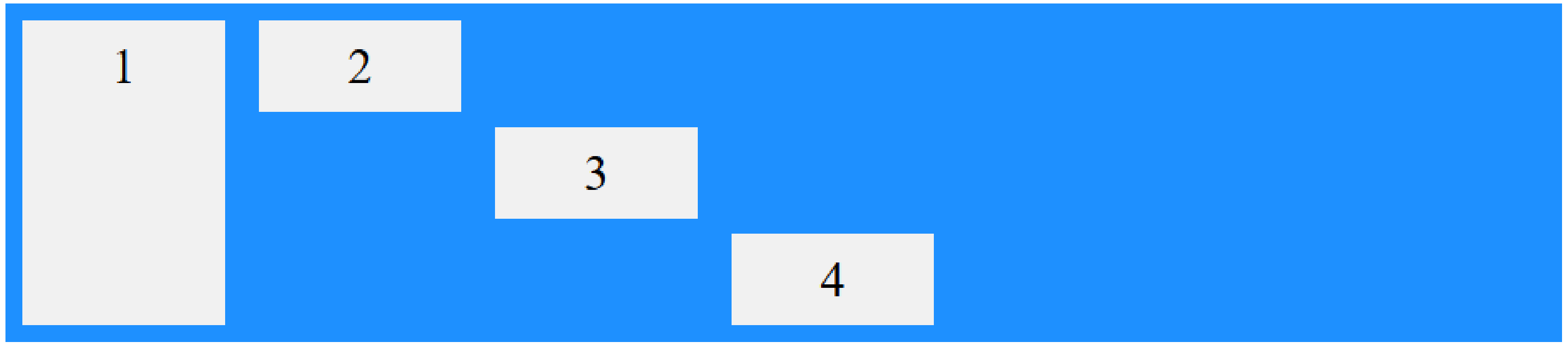
```
<div class="container">

  <div>1</div>

  <div>2</div>

  <div style="flex-shrink: 0">3</div>

  <div>4</div>

  <div>5</div>

</div>
```



align-self - specifies the alignment for the selected item inside the flexible container.

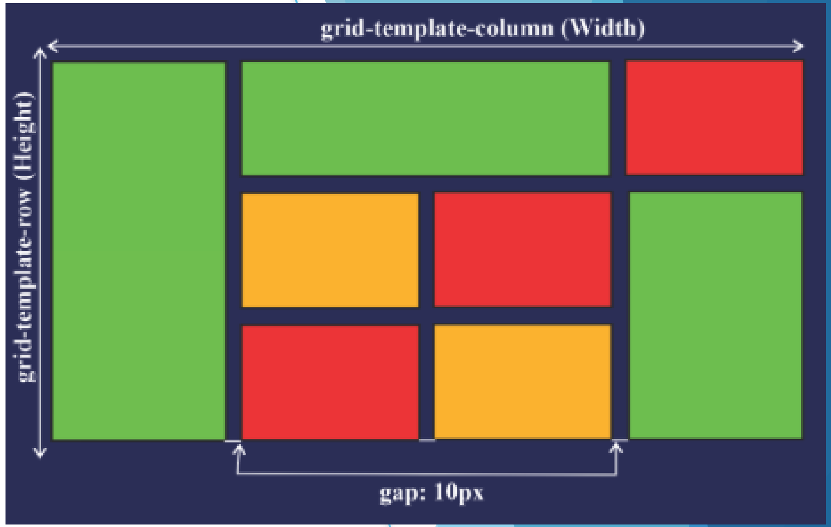
```
<style>
.container {
  display: flex; height: 200px;
  background-color: dodgerblue; }
.container div {
  background-color: #f1f1f1;
  color:#000;
  width: 100px;
  margin: 10px;
  padding: 10px;
  text-align: center;
  font-size: 30px; }
</style>
```

```
<div class="container">
  <div>1</div>
  <div style="align-self: flex-start">2</div>
  <div style="align-self: center">3</div>
  <div style="align-self: flex-end">4</div>
</div>
```



CSS – Grid

is a two-dimensional grid-based layout system that, compared to any web layout system of the past, completely changes the way we design user interfaces.



- **Difference Between CSS Grid and Flexbox**

Property	Grid	Flexbox
Dimension	Two - Dimensional	One - Dimensional
Features	Can flex combination of items through space-occupying Features	Can push content element to extreme alignment
Support Type	Layout First	Content First
Primary Use Case	Creating complex layouts with rows and columns	Aligning items in a row or column

- **CSS Grid properties**

(Grid Container)

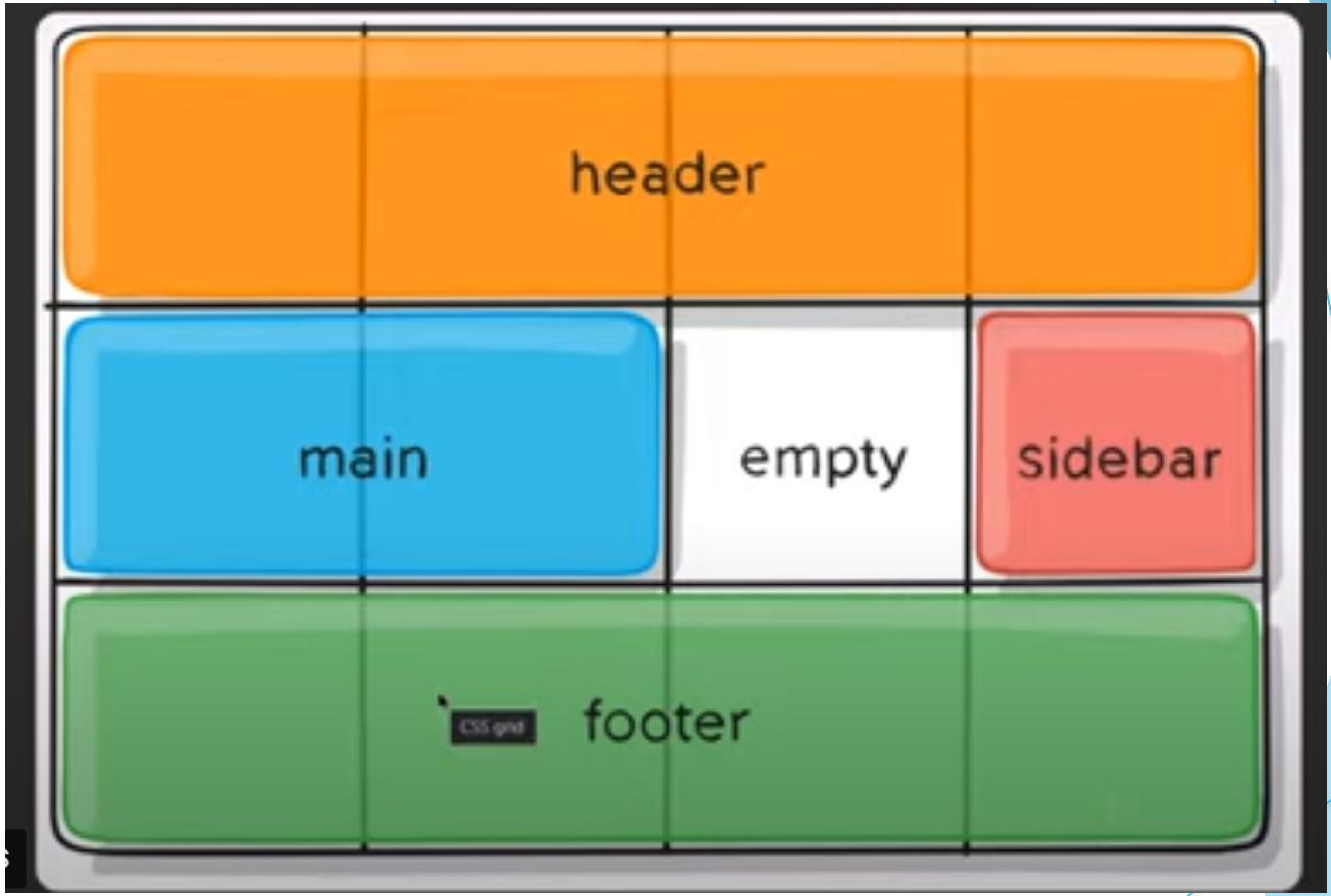
Display

grid-template-columns	grid-template-rows
column-gap	row-gap
gap	
justify-content	align-content
grid-template-areas	

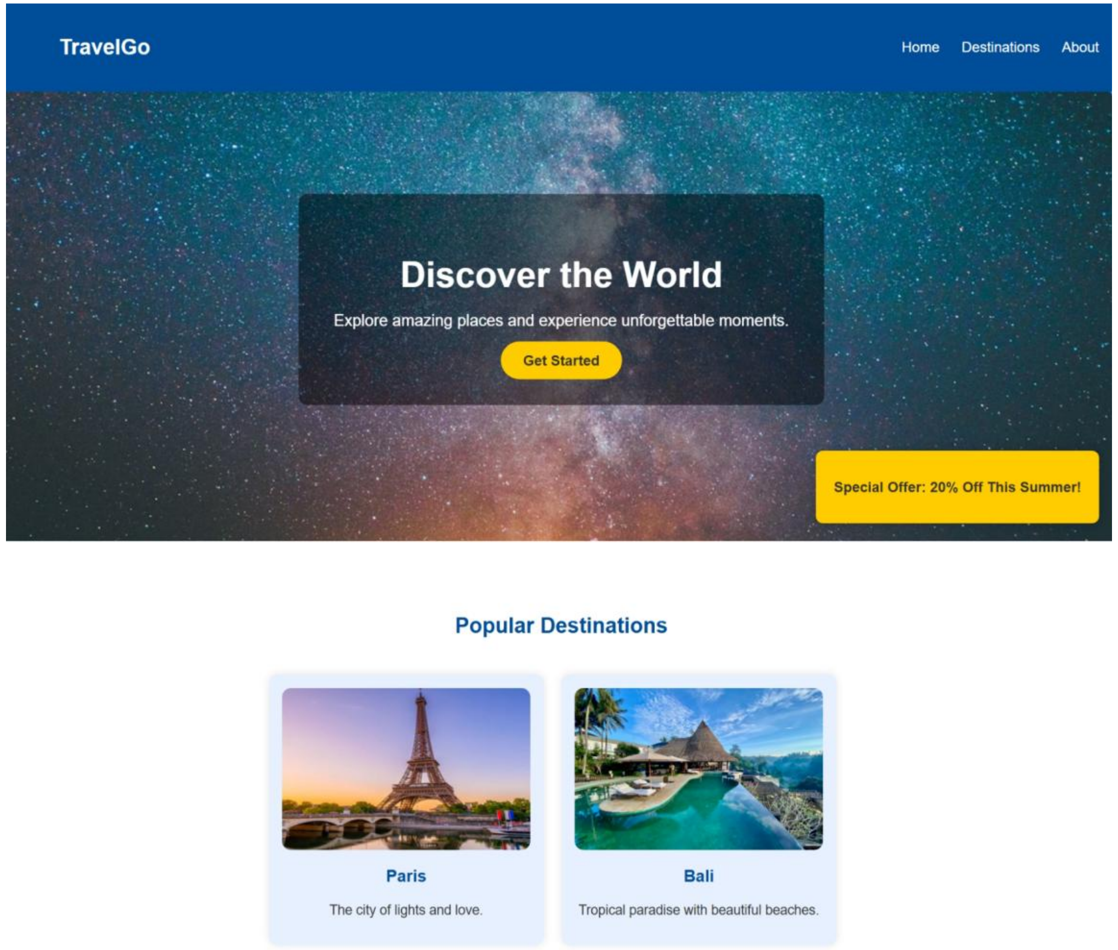
(Grid Items)

grid-column	grid-row
grid-area	

- **CSS Grid Example**



Assignment


TravelGo

Home Destinations About

Discover the World

Explore amazing places and experience unforgettable moments.

Get Started

Special Offer: 20% Off This Summer!

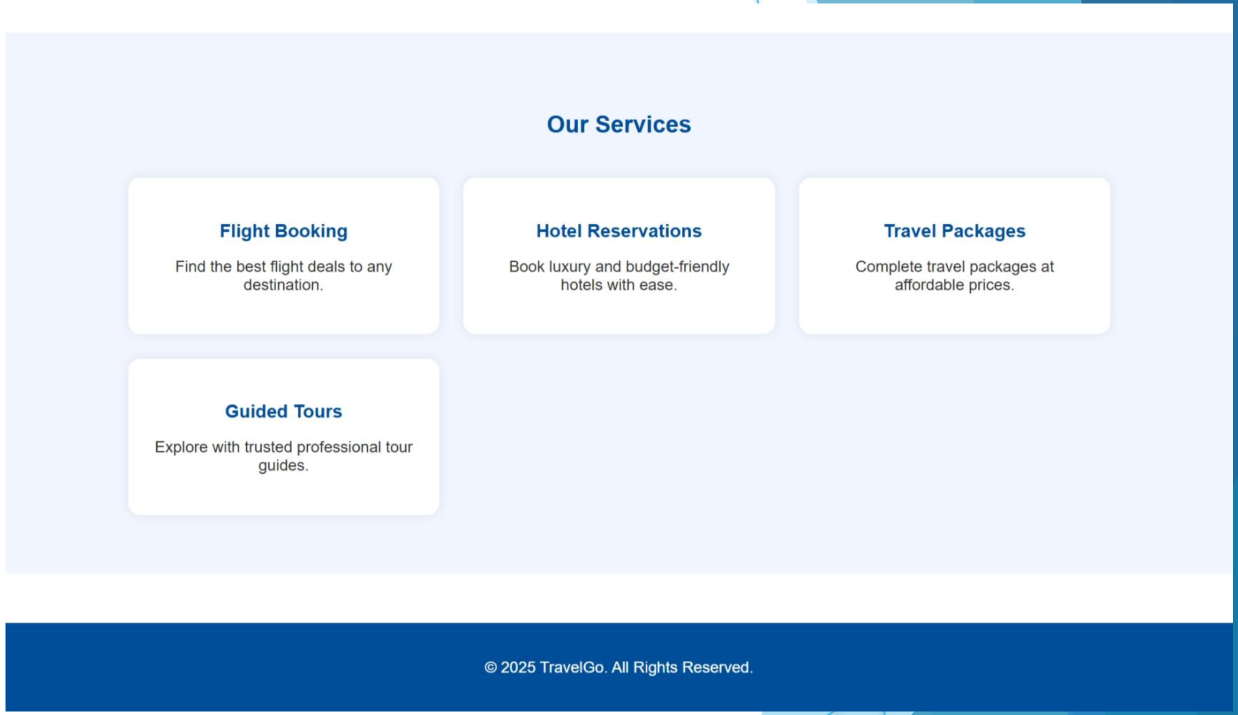
Popular Destinations

Paris

The city of lights and love.

Bali

Tropical paradise with beautiful beaches.


Our Services

Flight Booking

Find the best flight deals to any destination.

Hotel Reservations

Book luxury and budget-friendly hotels with ease.

Travel Packages

Complete travel packages at affordable prices.

Guided Tours

Explore with trusted professional tour guides.

© 2025 TravelGo. All Rights Reserved.

CSS – Transitions

allows you to change property values smoothly, over a given duration.

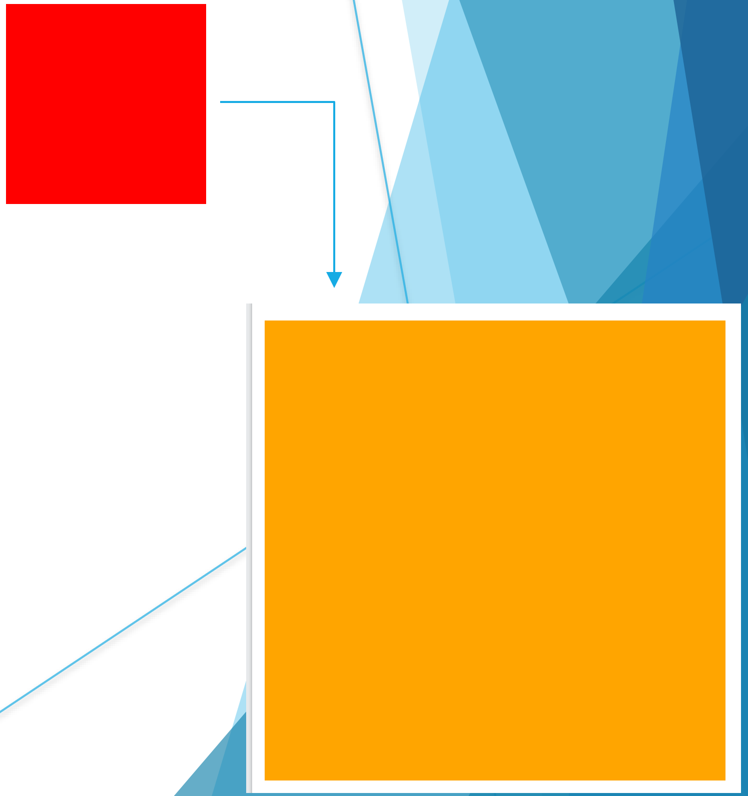
The CSS transition property is a shorthand property for:

- transition-property (Required)
- transition-duration (Required)
- transition-timing-function
- transition-delay

- **transition-property** is triggered when there is a change in the element's properties. This often happens within pseudo-classes (:hover, :active, :focus, or :checked).

- You can change multiple properties by separating them by commas.

```
<style>
div {
  width: 100px; height: 100px;
  background-color: red;
  transition: width 2s, height 4s, background-color 3s; }
div:hover {
  width: 300px;   height: 300px;  background-color: orange; }
</style>
<div></div>
```



- **transition-timing-function** specifies the speed curve of the transition effect.

This property can have one of the following values:

- **ease** - transition will start slow, then go fast, and end slow (this is default)
- **linear** - transition will keep the same speed from start to end
- **ease-in** - transition will start slow
- **ease-out** - transition will end slow
- **ease-in-out** - transition will have a slow start and end

---

- **transition-delay** specifies a delay before the transition starts.

value is defined in seconds (s) or milliseconds (ms).

CSS – Animations

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times as you want.

To use CSS animation, you must specify some keyframes for the animation.

- **animation-name & animation-duration**

  - animation-name property specifies a name for the animation.

  - animation-duration property defines how long an animation should take to complete.
- **@keyframes Rule**
- **animation-delay** property specifies a delay for the start of an animation.
- **animation-iteration-count** property specifies the number of times an animation should run.

CSS – Animations

- **animation-direction** property specifies whether an animation should be played forwards, backwards or in alternate cycles.

The animation-direction property can have the following values:

- **normal -** The animation is played as normal (forwards). This is default
- **reverse -** The animation is played in reverse direction (backwards)
- **alternate -** The animation is played forwards first, then backwards
- **alternate-reverse -** The animation is played backwards first, then forwards

CSS – Animations

```
<style>
div {
  width: 100px;   height: 100px;
  background-color: red;
  animation-name: myAnimation;
  animation-duration: 4s;   }
@keyframes myAnimation {
  from {background-color: red;}
  to {background-color: yellow;}
}
</style>

<div></div>
```

CSS – Responsive design

Responsive web design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge a website, to make it look good on all devices.

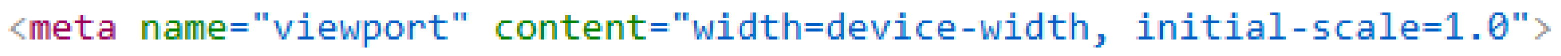
Key components in responsive web design are:

- Viewport <meta> tag
- Flexible layout (grid and flex)
- Media queries

- **The Viewport**

is the user's visible area of a web page.

varies with the device (will be a lot smaller on a mobile phone than on a computer screen).



```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

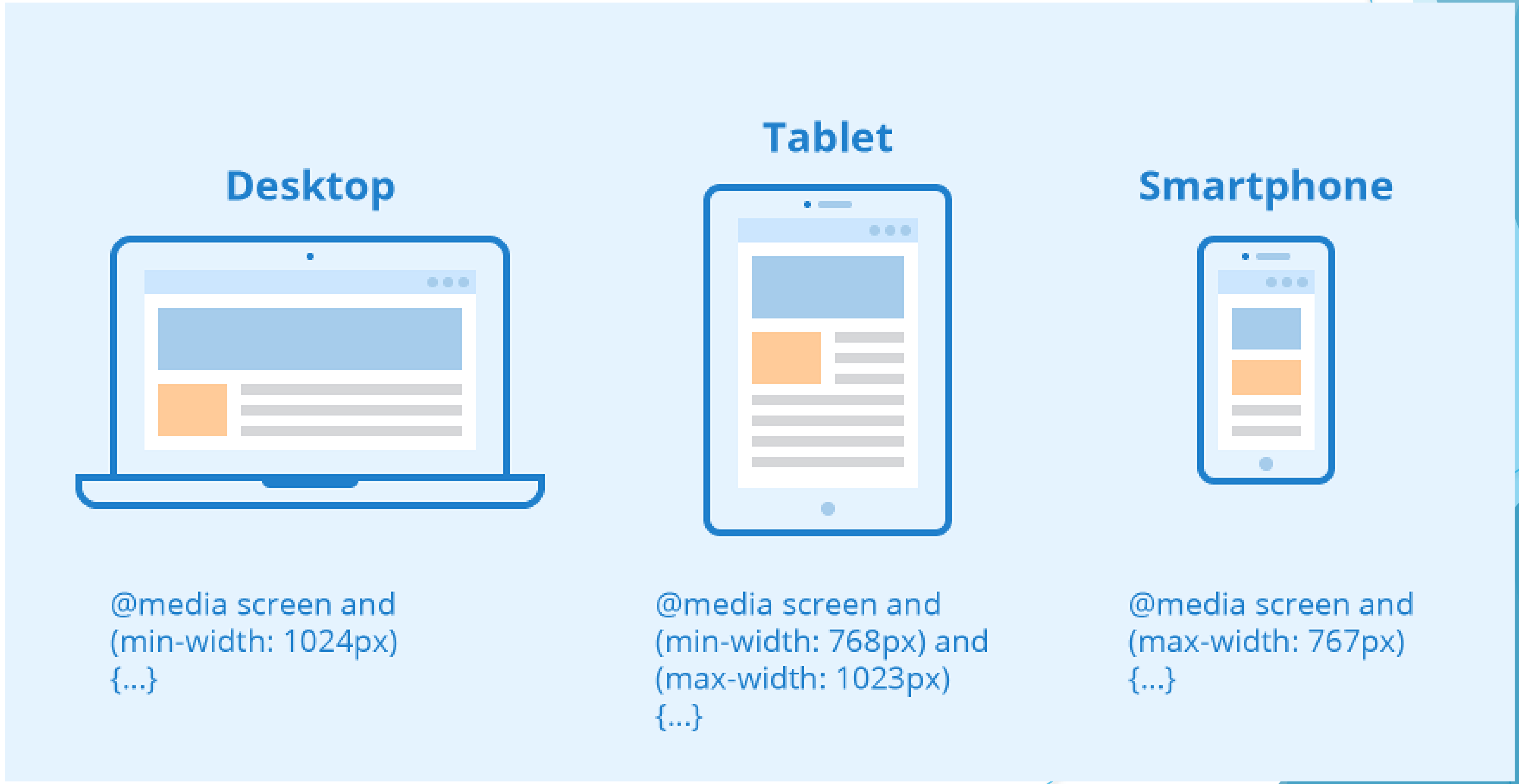
initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

- **Grid-View** means that the page is divided into rows and columns.

- **Media Queries** allow you to apply styles based on the characteristics of a device or the environment displaying the web page.

  CSS media queries are essential for creating responsive web pages.

  The CSS @media rule is used to add media queries to your style sheet.



```
@media screen and
(min-width: 1024px)
{...}
```

```
@media screen and
(min-width: 768px) and
(max-width: 1023px)
{...}
```

```
@media screen and
(max-width: 767px)
{...}
```

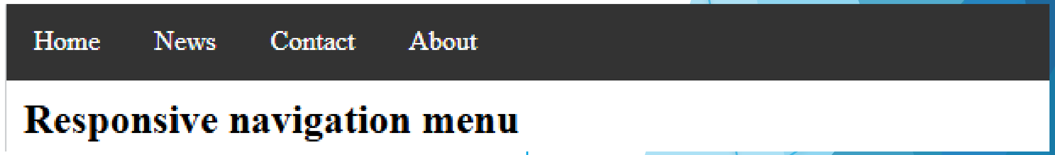
- **Media Query Syntax**

```
@media screen and (min-width: 480px) {
 /* CSS rules to apply */
}

@media screen and (min-width: 480px) and (max-width: 768px) {
 /* CSS rules to apply */
}
```

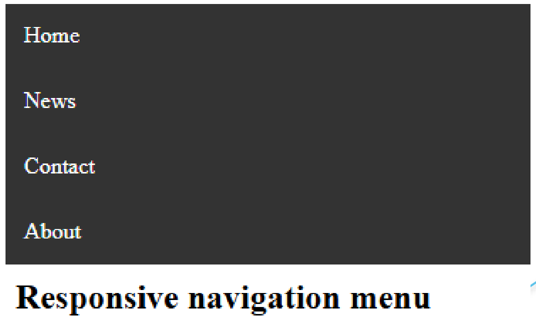
- **Media Query Example**

```
<style>
h2, p { margin: 10px; }
ul { list-style-type: none;  margin: 0;  padding: 0;
  background-color: #333333;  display: flex;  }
ul li a {
  display: block;  color: white;
  padding: 14px 16px;  text-decoration: none;  }
ul li a:hover { background-color: #111111; }
@media screen and (max-width: 600px) {
  ul {flex-direction: column;}
}
</style>
```

```
<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>
<h2>Responsive navigation menu</h2>
```


Home News Contact About

Responsive navigation menu


Home

News

Contact

About

Responsive navigation menu