CSS: Positions and Display

Lec-4

Positions in CSS

- As we know, block level elements occupy the complete width of an browser window and they also prevent other elements from appearing in same horizontal space.
- This is called as default position of an element and can be changed by setting it's position property.

> If we favor the default the default position of an HTML element, then we don't need to set it's position property.

- The position property can take one of the following values:

1. **static (default)**
2. **relative**
3. **absolute**
4. **fixed**
5. **sticky**

Positions in CSS

1. **Static (default):**

- All HTML elements are positioned static by default.
- Static positioned elements are not affected by the **offset properties like- top, bottom, left, right**.
- An element with `position: static;` is always positioned according to the normal flow of the page.

2. **Relative:**

- `position: relative;` allows you to position an element **relative** to it's **default static position on the web page.**
- This is done by accompanying the position declaration with one or more of the following offset properties that will move elements from its default static position.
- Following are the offsets properties which shall be **compulsorily given in case of position: relative**:

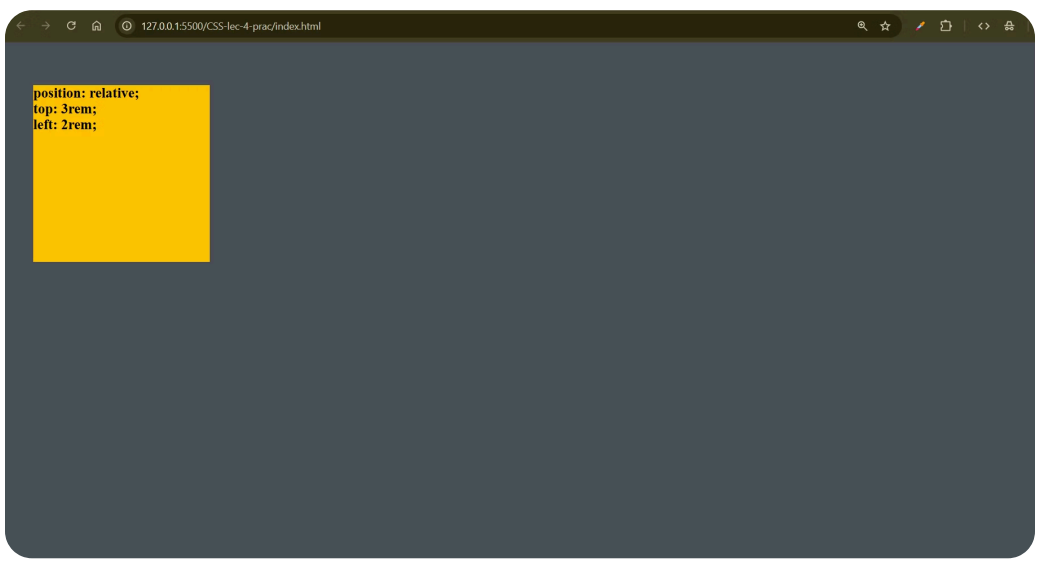
(i) top: moves the element down from top.

(ii) bottom: moves the element up from bottom.

(iii) left: moves the element away from the left side **(towards right)**.

(iv) right: moves the element away from the right side **(towards left).**

before position: relative



after position: relative

Positions in CSS

3. **Absolute:**

- `position: absolute;` allows an element to be **positioned relative to its closest ancestor** which has it's **position property set to anything other that static (can be relative, absolute or fixed).**
- After using `position: absolute;` **offset properties can be used to adjust its final position from that reference point.**

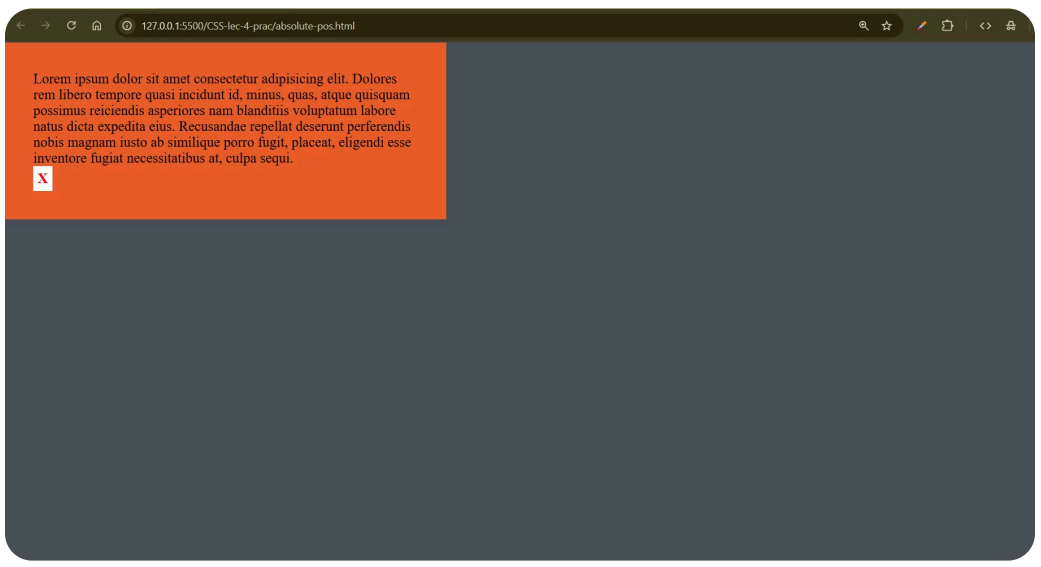
Eg:

HTML

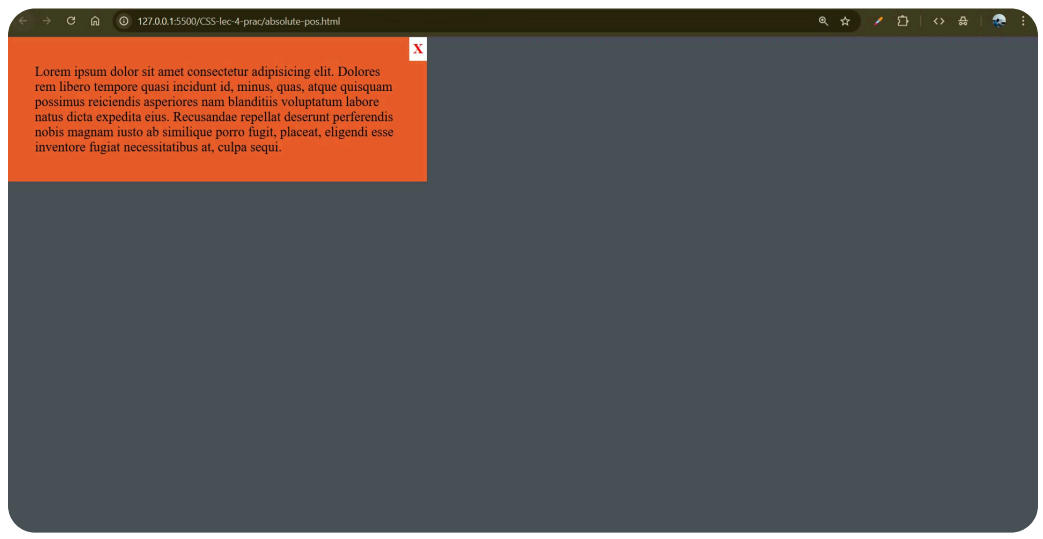
```
<div class="parent-cont">
  Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolores rem libero tempore quasi incidunt id, minus, quas, atque quisquam possimus reiciendis asperiores nam blanditiis voluptatum labore natus dicta expedita eius. Recusandae repellat deserunt perferendis nobis magnam iusto ab similique porro fugit, placeat, eligendi esse inventore fugiat necessitatibus at, culpa sequi.

<div class="child-cont"> X </div>

</div>
```

CSS

```
.parent-cont{
    background-color: #eb5e28;
    width: 500px;
    padding: 2rem;
    position: relative;
}
.child-cont{
    width: fit-content;
    padding: 5px;
    background-color: white;
    color: red;
    font-weight: bold;
    cursor: pointer;
    position: absolute;
    top:0;
    right:0;
}
```



before position: absolute



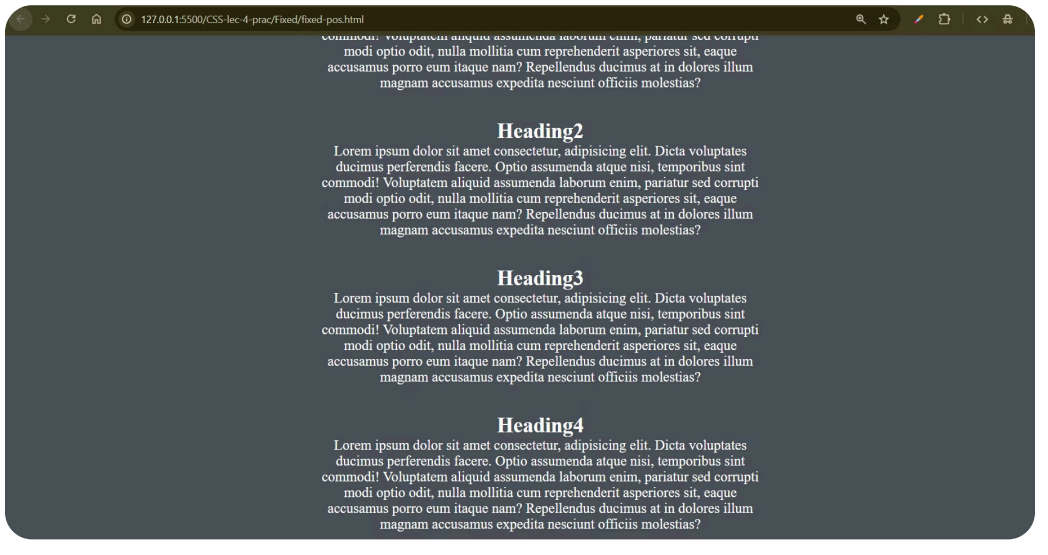
after position: absolute

Positions in CSS

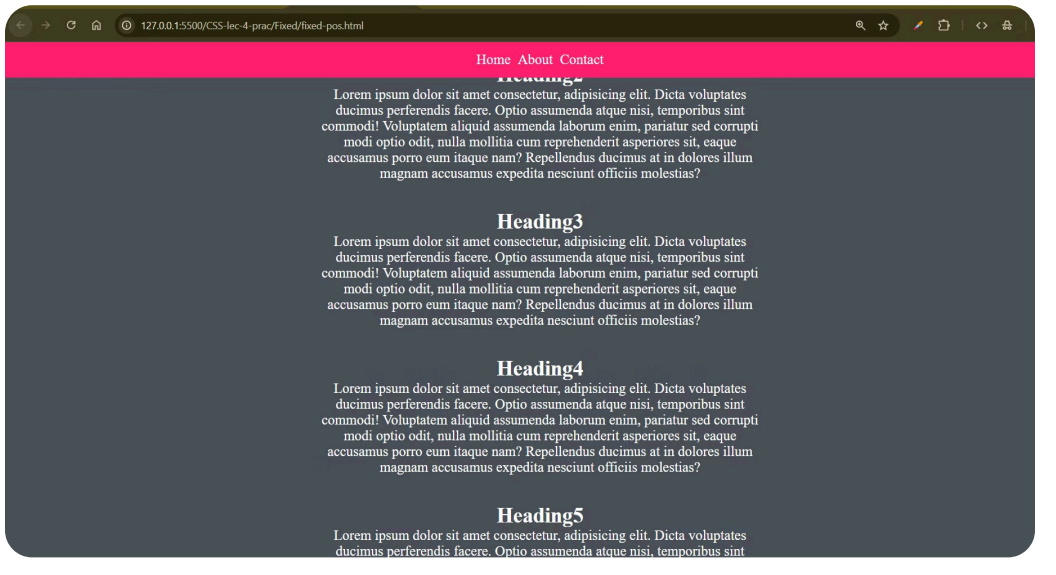
4. **Fixed:**

- We can **fix an element to a specific position on the page (regardless of user scrolling)** by setting it's position to **fixed**, and accompanying it with the familiar offset properties i.e **top, bottom, left and right.**

Eg:



before position: fixed



before position: fixed

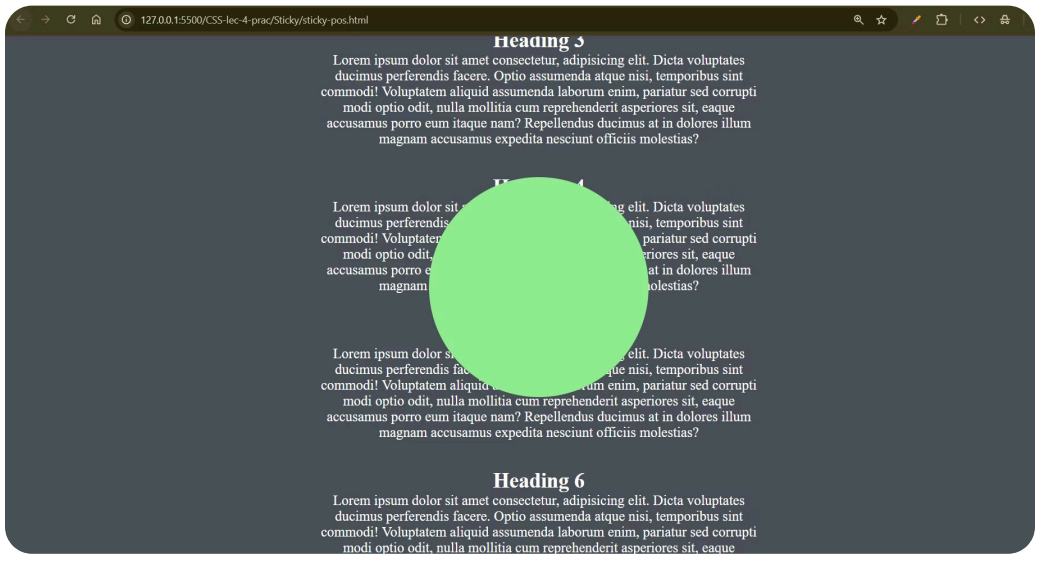
- This technique is often used for **navigation bars** on a web page.

5. **Sticky:**

- `position: sticky;` keeps an element in the document flow as the user scrolls, b**ut sticks to a specified position as the page is scrolled further.**
- This is done by using the **sticky** value along with the **familiar offset properties**, as well as new one.

Eg:

before position: sticky



after position: sticky

Z- Index

- When boxes on a webpage have a **combination of different positions, the boxes (and therefore their content) can overlap with each other**, making the content difficult to read.
- The **Z-Index** property **controls how far back or how far forward an element should appear on webpage when elements overlap.**
- The **Z-Index property accepts integer values,** depending on their values, **the integers instructs the browser on the order in which elements should be beyond on the web page.**
- To apply **Z-Index** property **the position of element should be anything apart from static.**
- **When we do not specify the Z-Index property of any element then it is by default taken as 0.**

Eg:

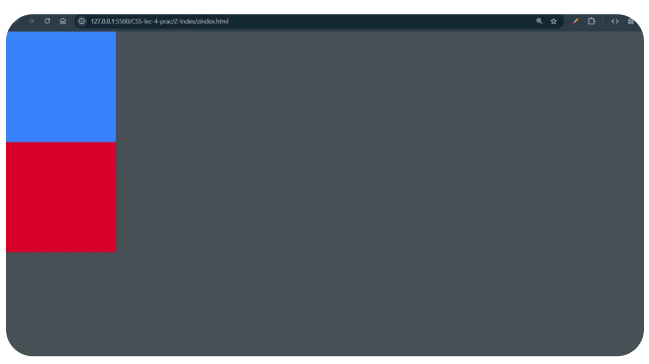
HTML

```
<div class="blue-box"</div>
<div class="red-box"></div>
```

CSS

```
.blue-box{
  background-color: #3a86ff;
  width: 200px;
  height: 200px;
}

.red-box{
  background-color: #d90429;
  width: 200px;
  height: 200px;
}
```

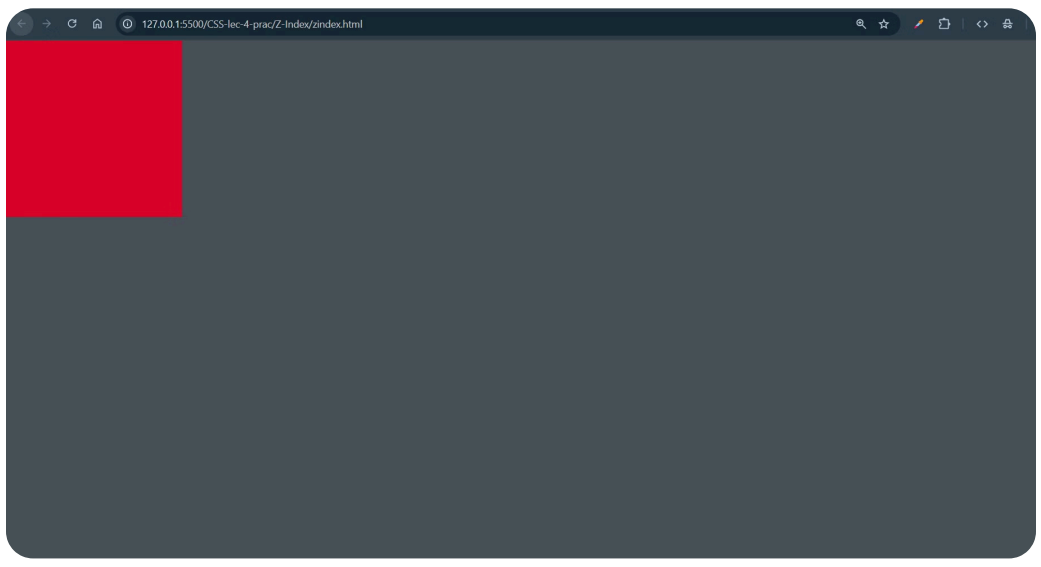
Output



CSS

```
.red-box{
    background-color: #d90429;
    width: 200px;
    height: 200px;
}
.red-box{
    background-color: #d90429;
    width: 200px;
    height: 200px;
    position: relative;
    bottom: 200px;
}
```

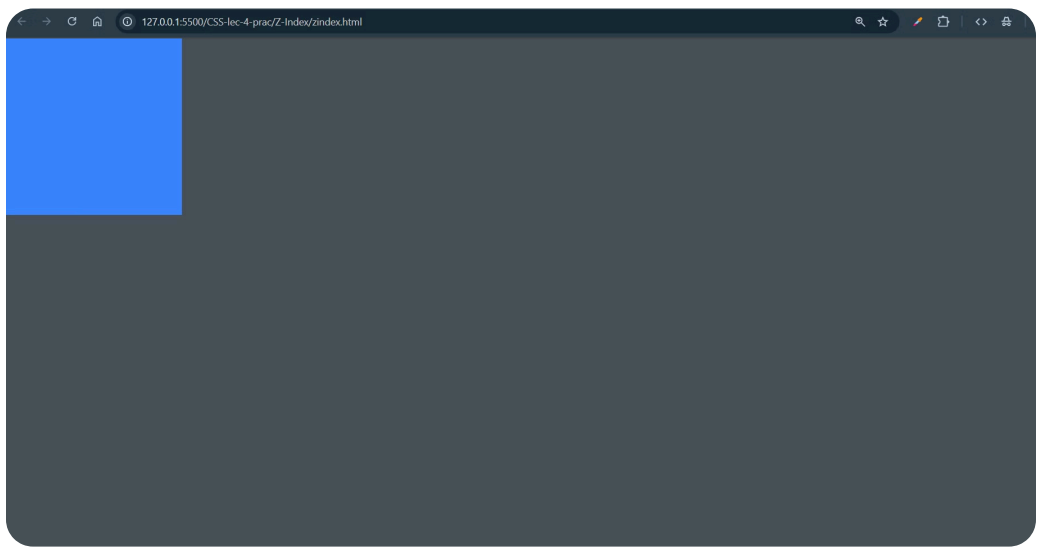
Output



CSS

```
.blue-box{
    background-color: #3a86ff;
    width: 200px;
    height: 200px;
    position: relative;
    z-index: 1;
}
.red-box{
    background-color: #d90429;
    width: 200px;
    height: 200px;
    position: relative;
    bottom: 200px;
}
```

Output



Display Property in CSS

a.) display: inline

- Every HTML element has a default **display** value that dictates if it can share horizontal space with other elements.
- In case of **inline elements,** they have a box which wraps tightly around their content occupying **only the width up to it's content.**
- The **display: inline** property provides the ability to make any element an **inline element**. This includes elements that are not inline by default such as **<p>, and headings.**

- The height and width of **inline element** neither can be specified nor can be altered in CSS.
- Eg of **inline element** - , <i>/, <u>/<ins>, <mark>, <del>, <sub>, <sup>, <q>, <small>, <big>, , <input>, <label>, <button>

- **Eg:**

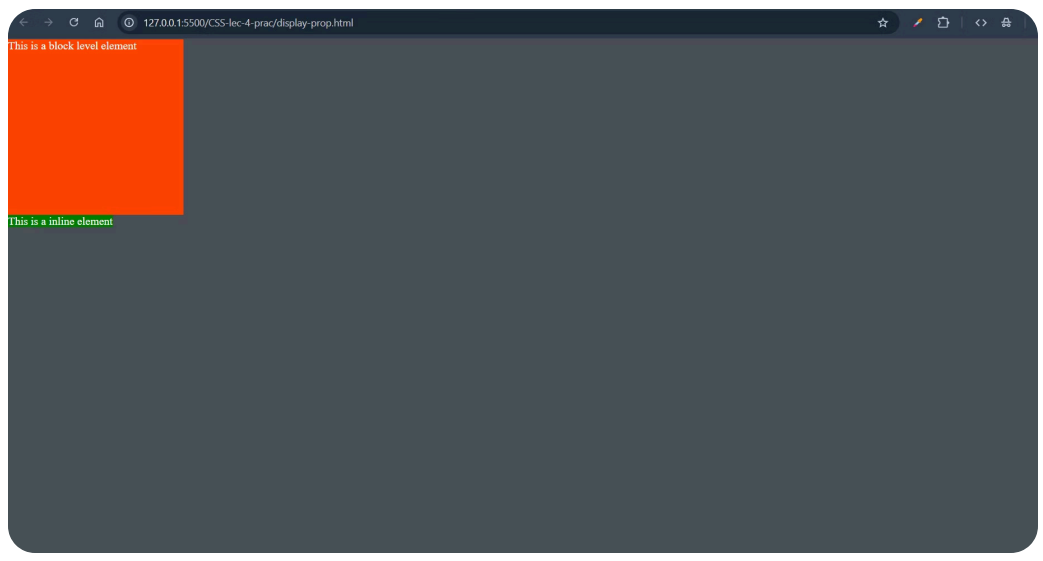
HTML

```
<body>
  <div>This is a block level element</div>
  <span>This is a inline element</span>
</body>
```

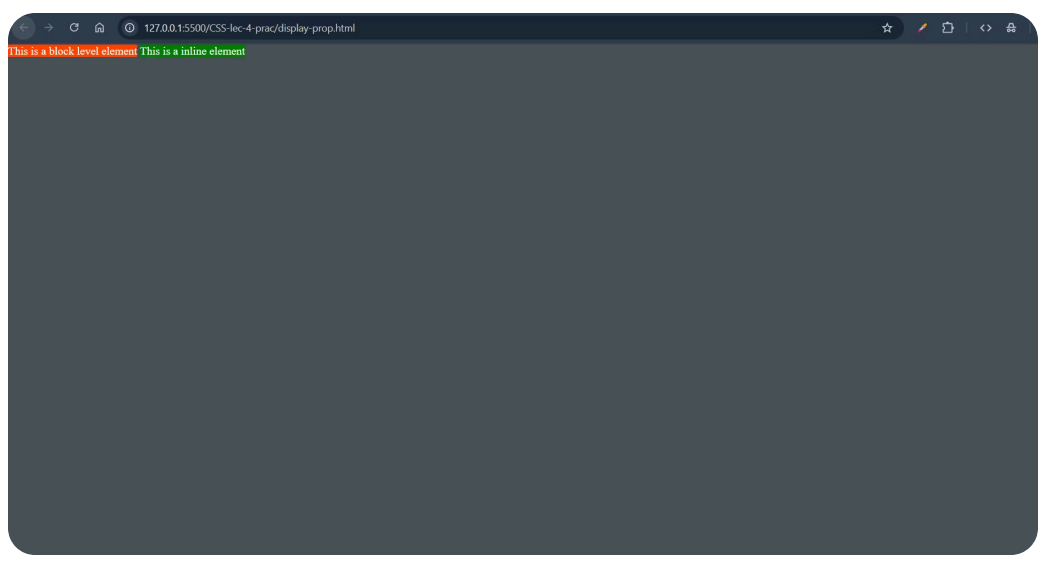
CSS

```
div {
  background-color: orangered;
  width: 250px;
  height: 250px;
}
span {
  background-color: green;
}
```

Output:




```
div {
  background-color: orangered;
  width: 250px;
  height: 250px;
  display: inline;
}
```



Display Property in CSS

b.) display: block

- Some elements are not displayed in same line as content around them, these are called **block-level elements.**
- The **display: block** property provides the ability to make any element an **block-level element**. This includes elements that are not inline by default such as **,<input>,etc.**

> - **Block-level** elements by default **occupies the entire width of the browser/web page.**
> - The **height** and **width** properties of these elements can be altered or changed.
> - Eg: ** , All Paired Tags, <p>, , <form>, semantic elements.**

- Eg:

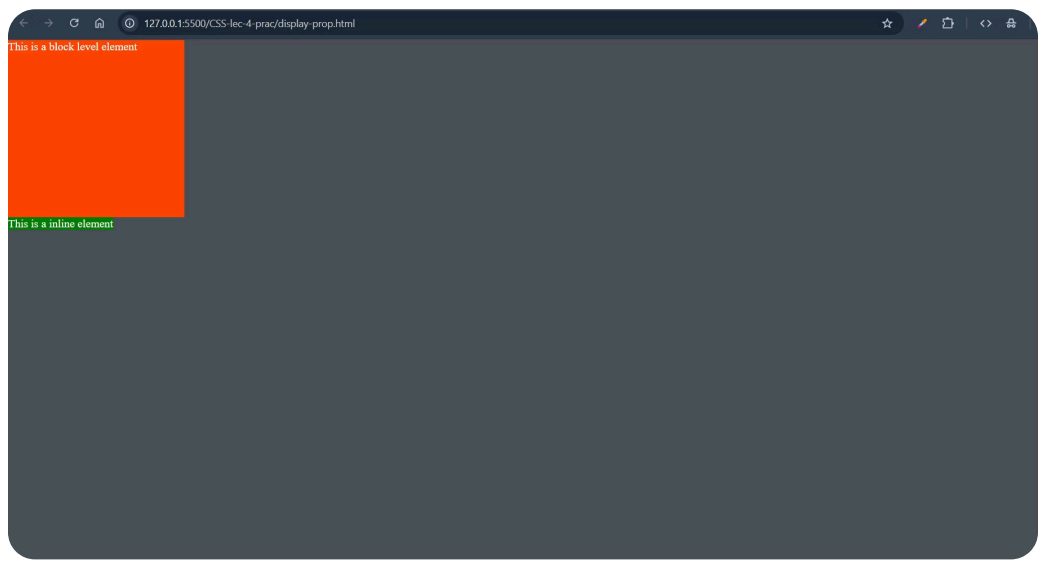
HTML

```
<body>
  <div>This is a block level element</div>
  <span>This is a inline element</span>
</body>
```

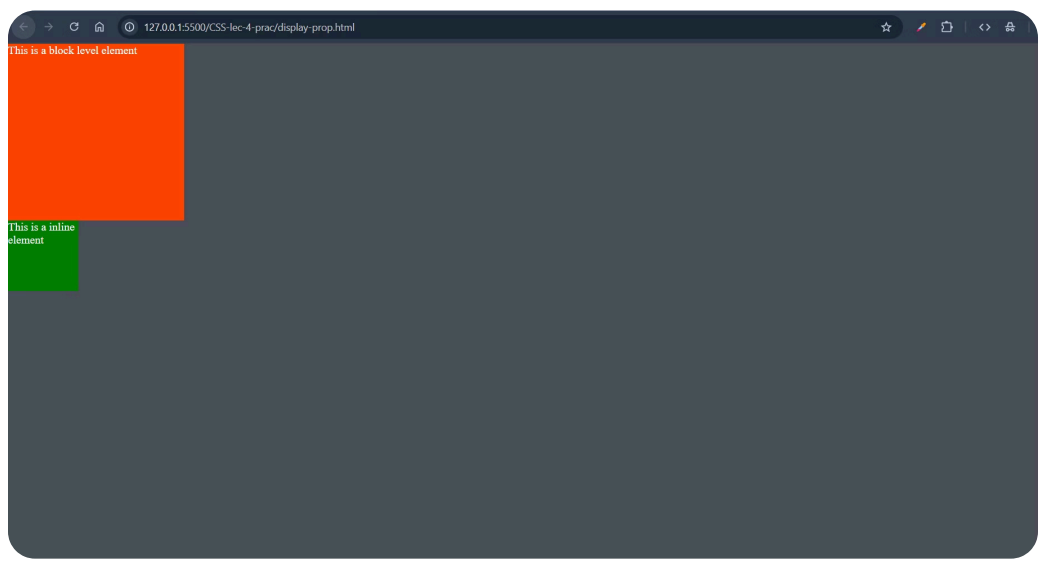
CSS

```
div {
  background-color: orangered;
  width: 250px;
  height: 250px;
}
span {
  background-color: green;
}
```

Output




```
span {
  background-color: green;
  display: block;
  width: 100px;
  height: 100px;
}
```



Display Property in CSS

c.) display: inline-block

- **Inline-block** display combines features of both **inline** and **block-level elements,**
- Inline-block elements can **appear next to each other only by occupying content width.**
- **We can specify height and width of these elements similar to block level elements.**
- Eg: **<img>, <audio>,<video>,<iframe>, etc.**

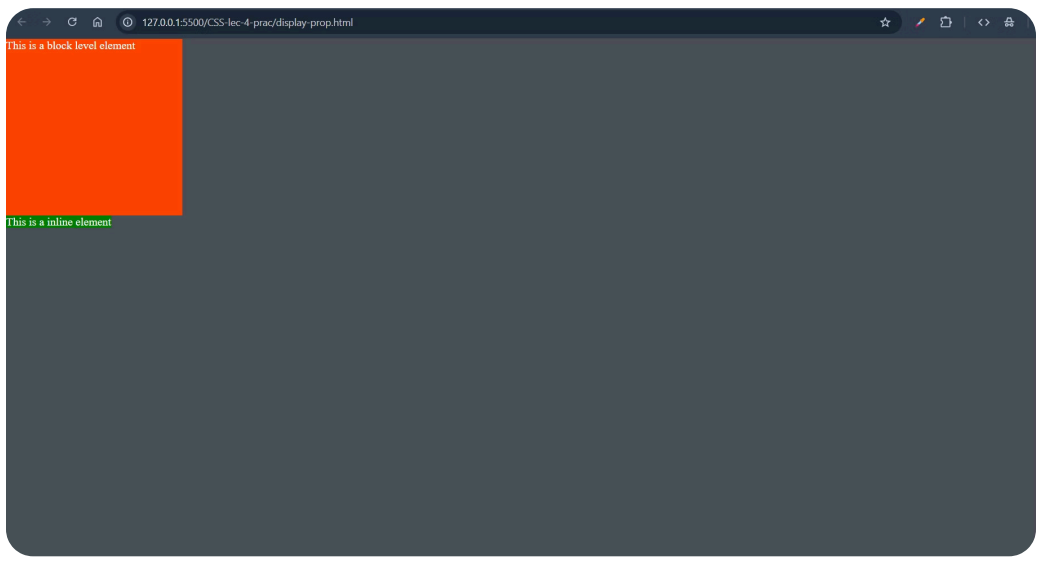
HTML

```
<body>
  <div>This is a block level element</div>
  <span>This is a inline element</span>
</body>
```

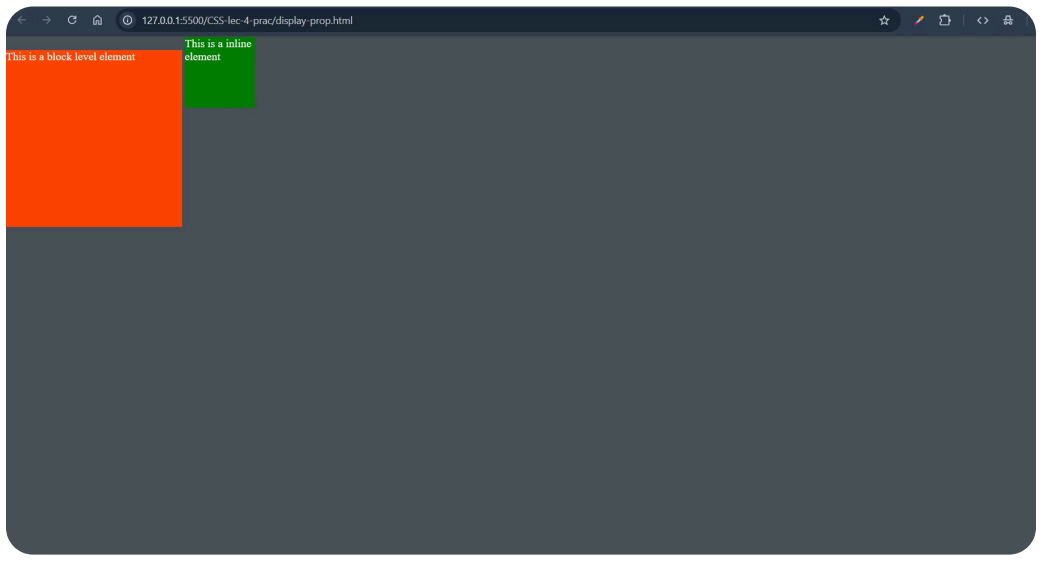
CSS

```
div {
  background-color: orangered;
  width: 250px;
  height: 250px;
}
span {
  background-color: green;
}
```

Output




```
div {
  background-color: orangered;
  width: 250px;
  height: 250px;
  display: inline-block;
}
span {
  background-color: green;
  display: inline-block;
  width: 100px;
  height: 100px;
}
```



Float Property In CSS

- In order to move an element **as far left as possible** or **as far right as possible** in the container, you can use the **float** property.
- The **float** property is **commonly used for wrapping text around an image.**
- The float property is often set using one of the values below:

1. **left:** moves or floats element as far left as possible.
2. **right:** moves or floats elements as far right as possible.

> - **Floated elements** always must have a **width specified**, otherwise the element **will assume full width of it's containing element** and changing float value will not yield any visible results.

- Eg:

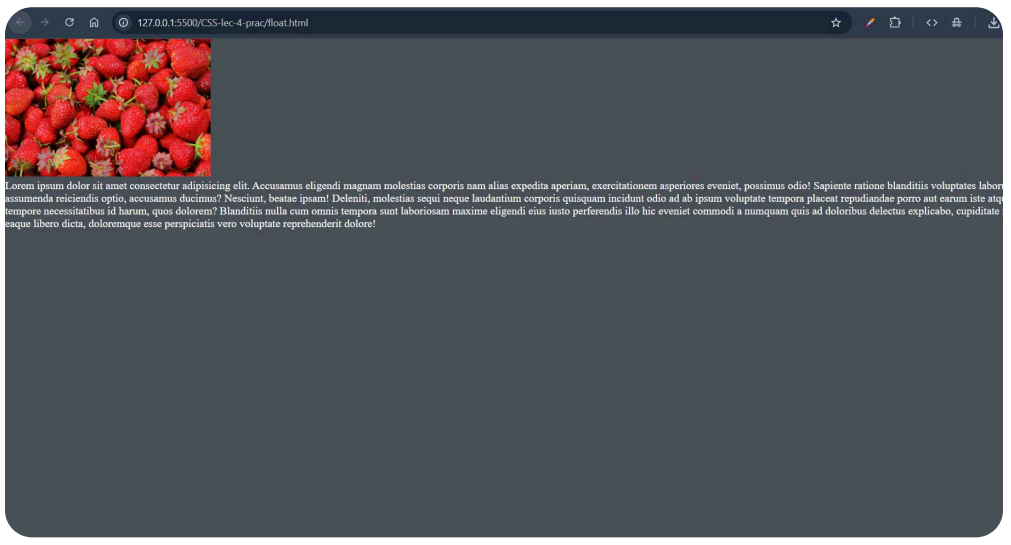
HTML

```
<div class="container">
    <img
      src="./Z-Index/fruit-img.jpg"
      alt="fruit-img"
      width="300px"
      height="200px"
    />
    <p>Lorem ipsum dolor sit amet consectetur
adipisicing elit. Accusamus
eligendi magnam molestias corporis nam alias
expedita aperiam </p>
</div>
```

CSS

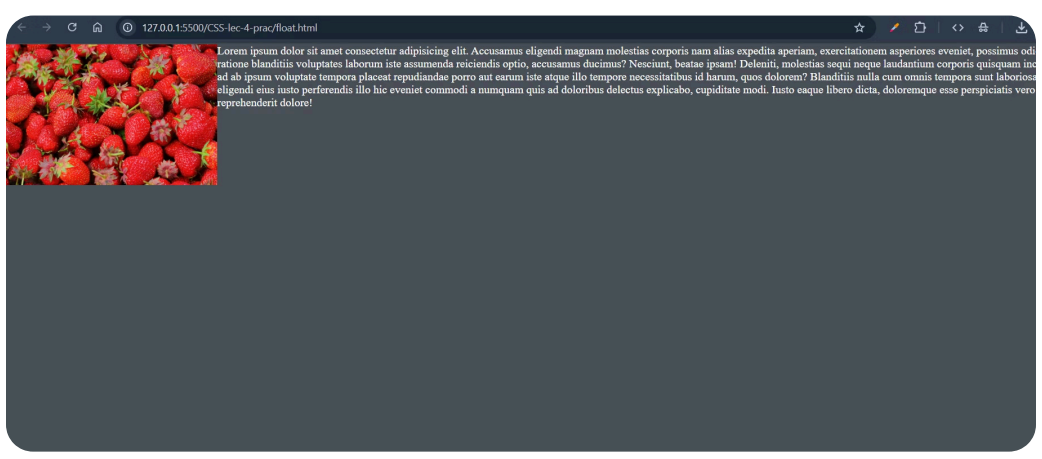
```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
body {
  background-color: #495057;
  color: white;
}
```

Output

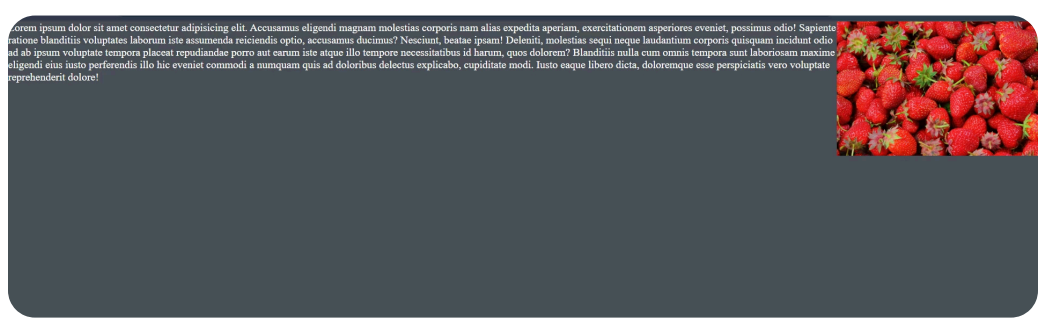


Change:

```
.container > img {
  float: left;
}
```



float: left



float: right

Clear Property In CSS

- The **clear** property controls the **flow next to floated elements.**
- The **clear** property specifies **what should happen with the element that is next to a floating element.**
- The **clear** property is often set using one of the values below:

Value	Description
none	The element is not pushed below left or right floated elements.
left	The element is pushed below left floated elements
right	The element is pushed below right floated elements
both	The element is pushed below both left and right floated elements

Eg:

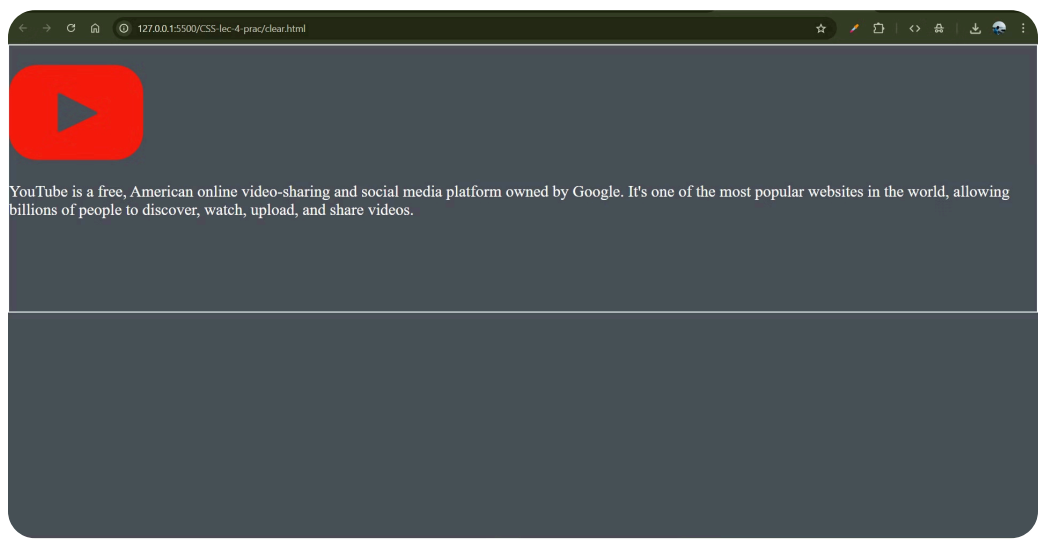
HTML

```
<div class="main-cont">

  <img src="./yt.png" alt="yt" />

   <p> YouTube is a free, American online video-sharing and social media platform owned by Google. It's one of the most popular websites in the world, allowing billions of people to discover, watch, upload, and sharevideos.
    </p>

</div>
```

CSS

```
div {
  border: 2px solid white;
  height: 400px;
}
img {
  width: 200px;
  height: 200px;
}
p {
  font-size: 1.5rem;
}
```

Output:



Change in CSS

```
img {
   float: right;
}
p {
  clear: none; // def value: makes no change
}
```

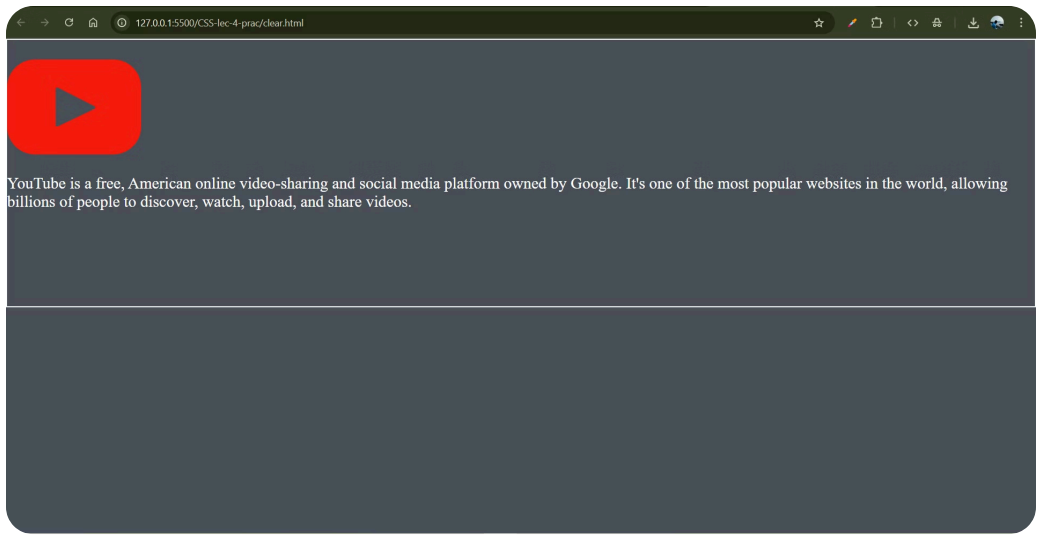
In the above example, the `none` (default value) of the `clear` property still allows the paragraph to wrap around the floated image element.

Clear Property In CSS

Clear- Left

```
img {
   float: left;
}
p {
  clear: left; // clears the left side and moves into a new
line
}
```

Output:



Clear- Right

```
img {
   float: right;
}
p {
  clear: right; // clears the left side and moves into a new
line
}
```

Output: