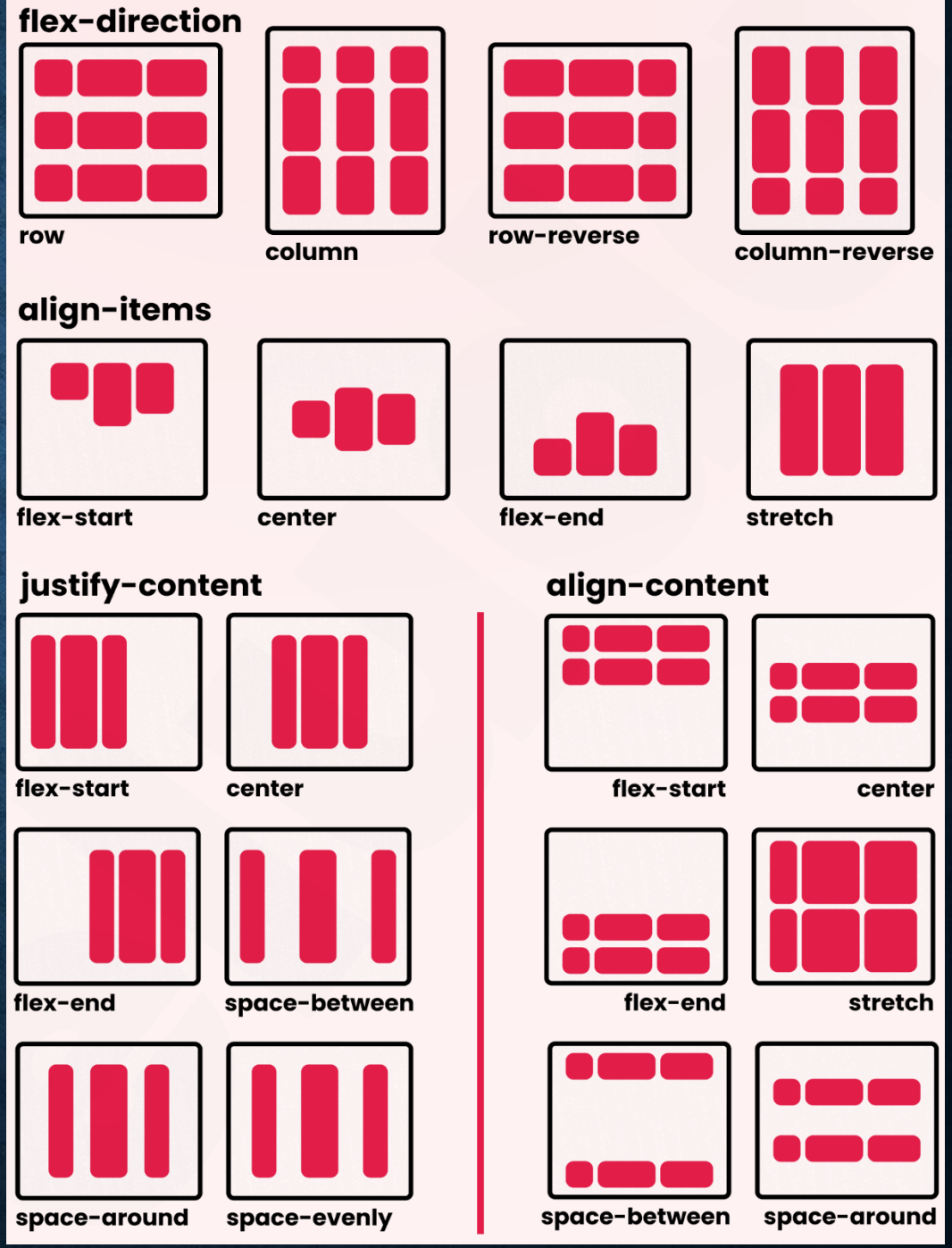
WEB TECHNOLOGY
LECTURE – 05
(CSS LAYOUT)

CSS Flexbox

FLEXBOX (FLEXIBLE BOX LAYOUT)

- The **Flexbox** layout is a CSS module designed to create one-dimensional layout with ease.
- To enable Flexbox, apply display: flex; to a container:



FLEXBOX (EXAMPLE)

```
.container {
  display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: wrap;
}
.item {
    width: 100px;
    height: 100px;
    background: lightblue;
    margin: 5px; display: flex;
    justify-content: center;
    align-items: center;
}
```


```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
</div>
```

THE ADVANCE USE OF DISPLAY

```
p{
  display: none;
}
#myDiv:hover p{
  display: block; /* flex/inline/inline-block/others*/
}
```

You will encounter this design in some navbars. You hover element and some links appear.

HTML

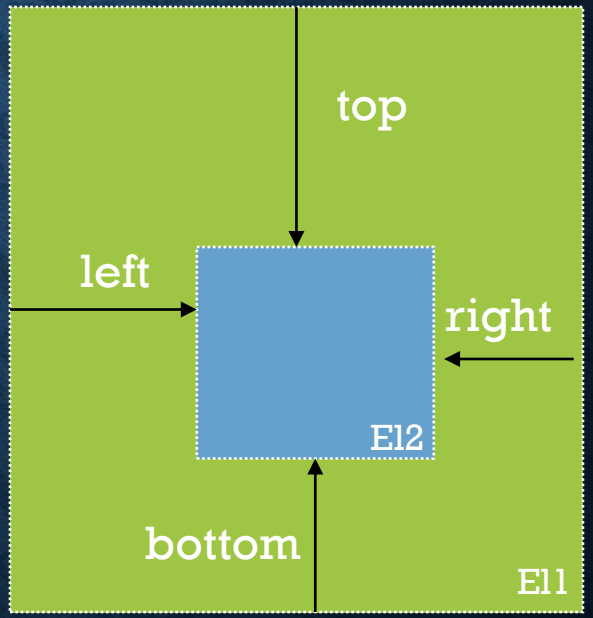
```
<div id="myDiv" style="background-color:rgb(175,200,156)">
    Hover me
    <p style="color:yellow"> Successful </p>
    <p style="color:green"> Congratulation </p>
</div>
```

CSS POSITIONING

CSS POSITIONING

- The last core concept to understand in CSS layout is positioning.
- There are five types of positioning that can be applied to CSS boxes:

1. **Static Positioning**
2. **Fixed Positioning**
3. **Relative Positioning**
4. **Absolute Positioning**
5. **Sticky Positioning**



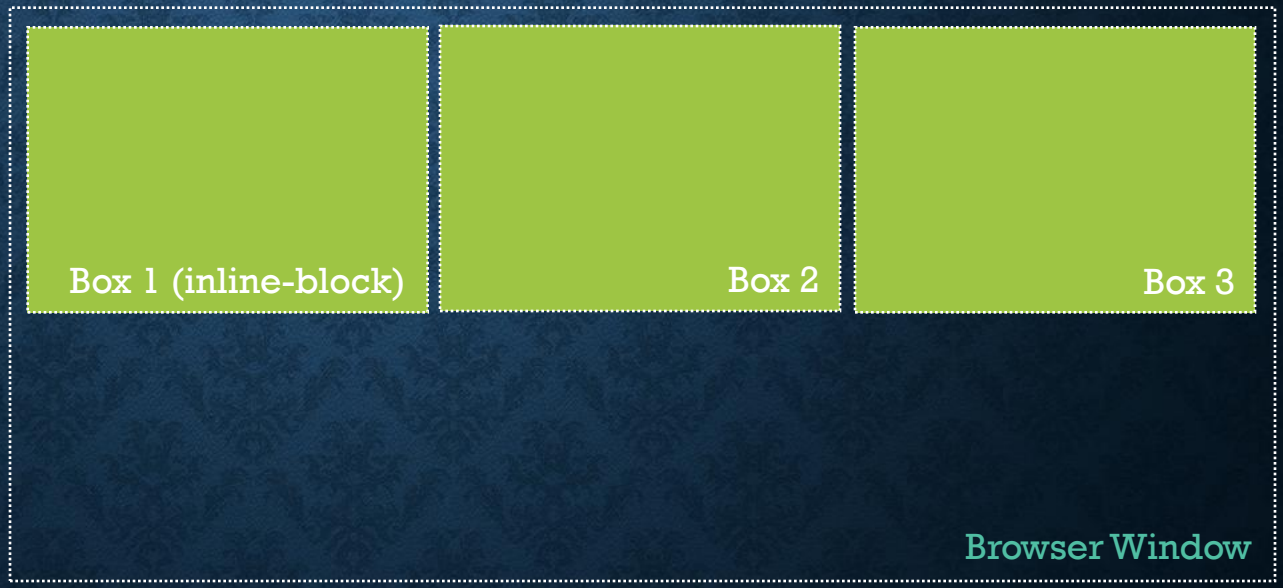
* top, bottom, left, and right are four properties related to positioning. These properties require a reference which vary with different type of positioning

STATIC POSITIONING

- HTML elements are positioned static by default.
- A static positioned element is always positioned according to the normal flow of the page.
- Static positioned elements are not affected by the top, bottom, left, and right properties.

```html
<div class="container">
    <div class="box1">Box1</div>
    <div class="box2">Box2</div>
    <div class="box3">Box3</div>
</div>
```

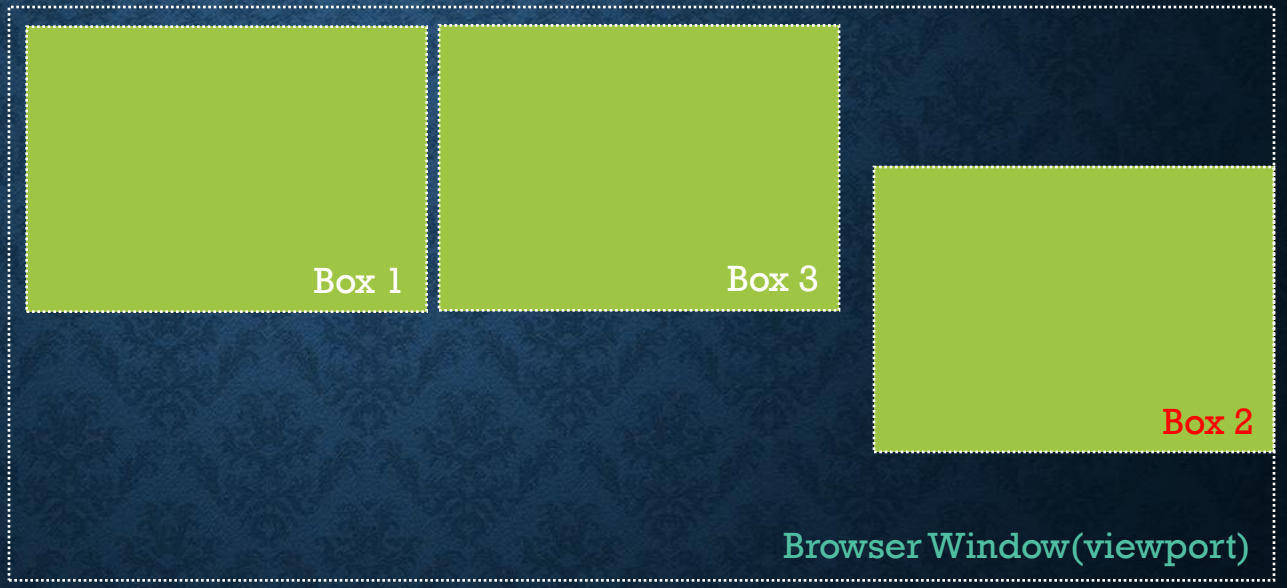
```css
.box1, .box2, .box3{
display: inline-block;
padding: 10px;
}
```



FIXED POSITIONING

- An element with fixed position is positioned relative to the browser window, i.e., viewport.
- It will not move even if the window is scrolled.

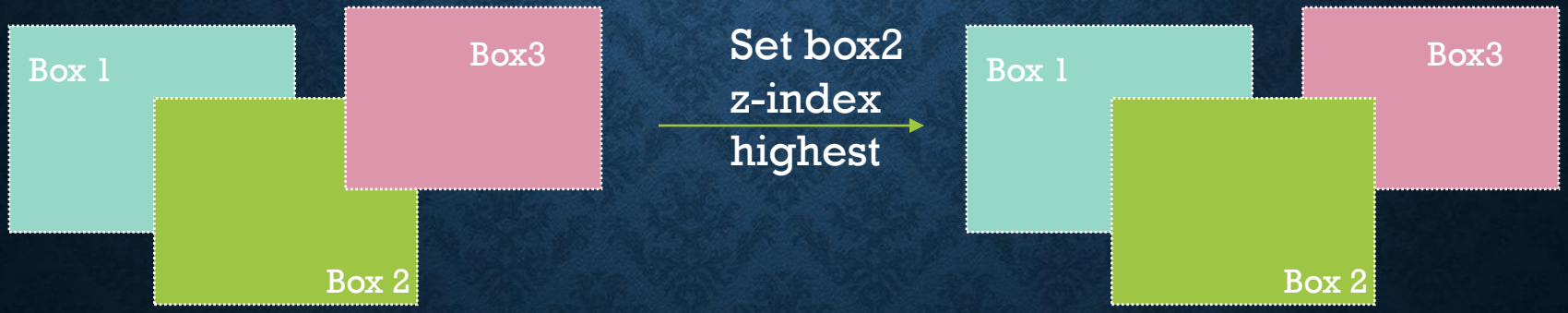
```
#box2 {
    position: fixed;
    right: 0px;
    top: 20px;
}
```



Notice that box3 has occupied the box2's normal flow position.

OVERLAPPING ELEMENTS

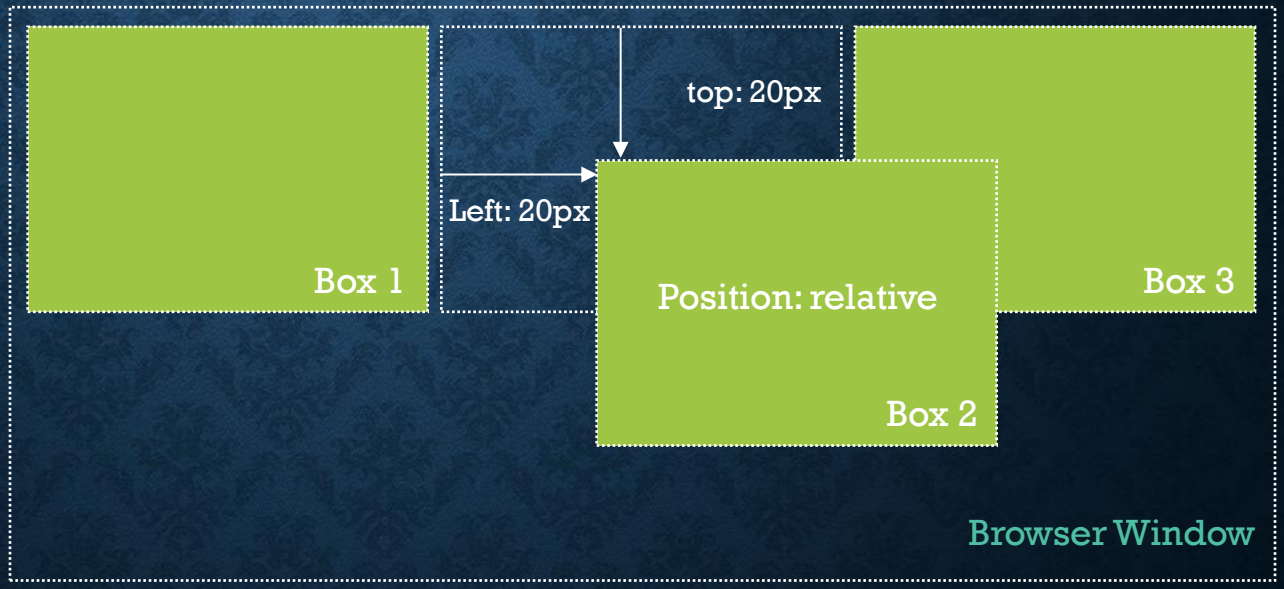
- When elements are positioned outside the normal flow, they can overlap other elements.
- The z-index property specifies the stack order of an element.



RELATIVE POSITIONING

- A relatively positioned element will position in relation to the normal flow.
- In this example, **box 2** is offset 20px, top and left. The result is the box is offset 20px from its original position in the normal flow. Box 2 may overlap other boxes in the flow, but other boxes still stay its original position in the flow.

```
#box2 {
    position: relative;
    left: 20px;
    top: 20px;
}
```

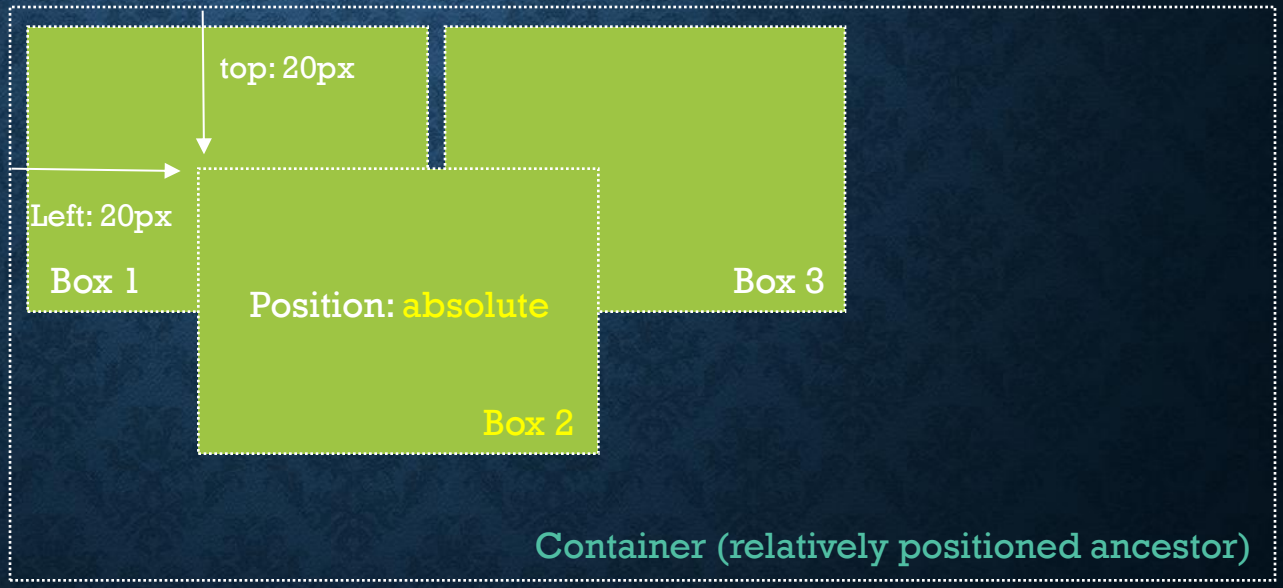


Notice that box3 does not occupy the box2's normal flow position.

ABSOLUTE POSITIONING

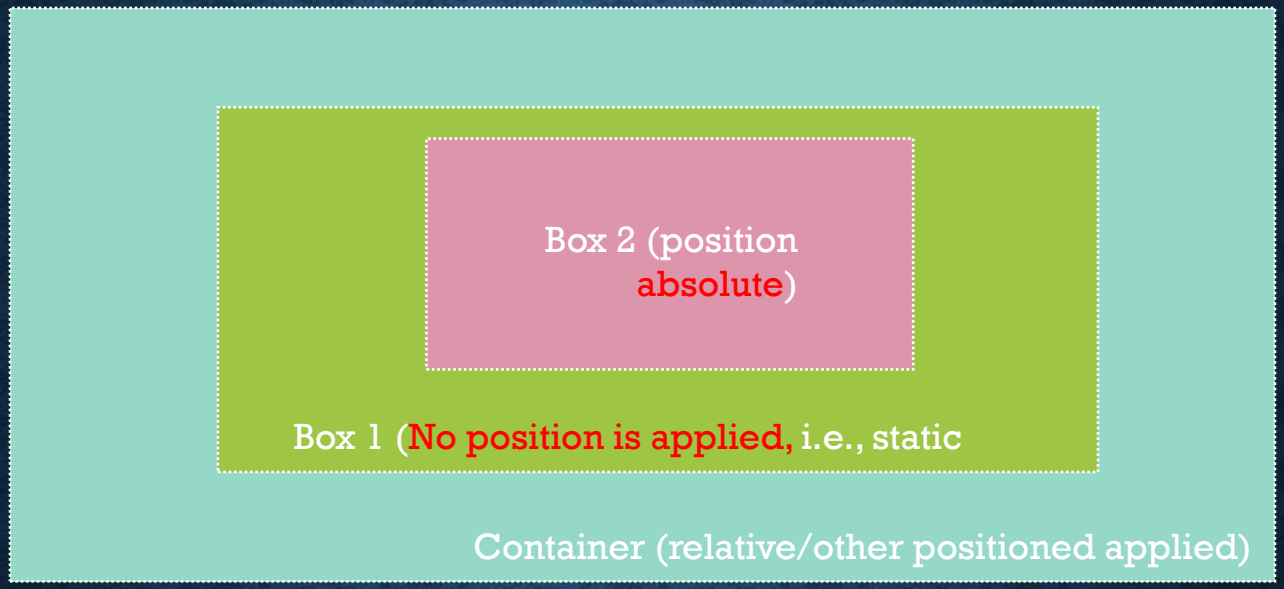
- An absolutely positioned box is taken out of the normal flow, and positioned in relation to its nearest positioned ancestor (i.e. its containing box).
- An element is called a "positioned element" when it has any value of the position property except static.
- If there is no ancestor box, it will be positioned in relation to the initial containing block, usually the browser window.

```
#box2 {
    position: absolute;
    left: 20px;
    top: 20px;
}
```



Notice that box3 has occupied the box2's normal flow position.

MORE ABOUT ABSOLUTE POSITIONING



- Although box1 is ancestor of box2, the top/left/right/bottom property will be affected by the container (not by the box1)
- This happens because box1 is not a positioned ancestor.

PERFECT CENTERING AN ELEMENT

```
#outer-box{
        width: 300px;       height: 300px;
        border: 1px solid black;
}
#inner-box {
    width: 100px;   height: 100px;
    position: absolute;
     left: 50%;     top: 50%; /* this 50% is measured on container size*/
     transform: translate(-50%, -50%);  /* this 50% is measured on its own size*/
    background-color: green;      text-align: center;
    padding: 20px;
    border: 2px solid black;
 }
```

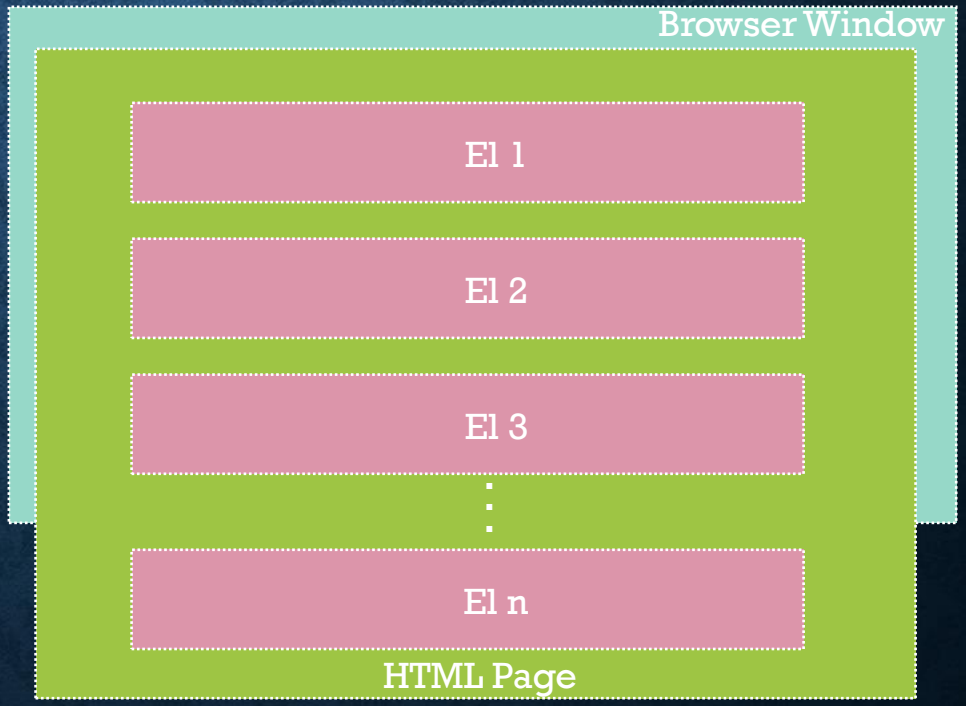
```
<div id="">
        <div id=""> </div>
</div>
```

STICKY POSITIONING

- It is used to prevent an element from getting out of container.
- Its top/left/right/bottom property activated only when it receive threads on its existence.

For example, If you to prevent El2 from being disappeared while scrolling down. Then you can set its position sticky and top to 0px;

However, if the container get out of window the sicky element would go as well.



RUN THE FOLLOWING CODE EXAMPLE

```
.container{
      height: 200px;
      width: 300px;
      margin: 20%;
      overflow: scroll;
   }
h1,h2,p{
      height: 50px;
      border: 1px solid black;}
h1{
      background-color: red;}
h2{
      background-color: green;}
p{
      background-color: blue; color: white;}
h2{
      position: sticky;
      top: 0px;
}
```

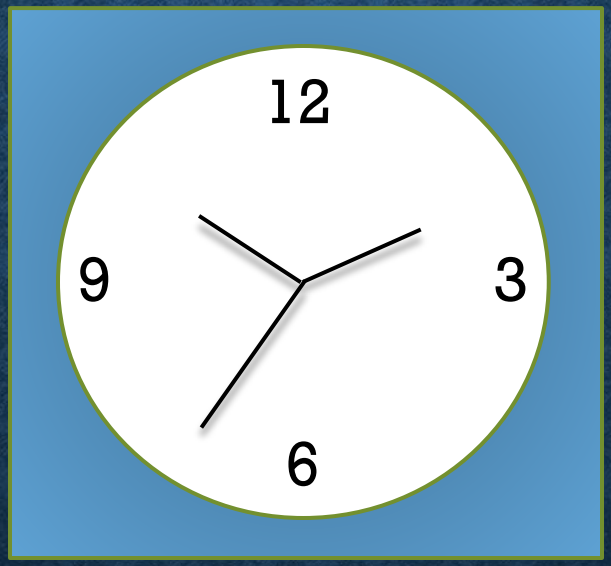
```
<div class="container">
      <h1>Hello</h1>
      <h2>Welcome to CSS</h2>
      <p>It is very conceptual tech</p>
      <p>It is very conceptual tech</p>
      <p>It is very conceptual tech</p>
      <p>It is very conceptual tech</p>
</div>
```

Tasks:

1. Add more element top of it and observe the scroll down behavior
2. Add more container below the container and observe the behavior

TASK

Can You Re-create the following design?



Hint: use multiple "-" and "*transform: rotate(5deg);*" property for the tick.

End of CSS

All slides are only for intended audience. Without consent sharing to others is prohibited.