CSS3 Table Properties

- When we build a HTML table without any styles or attributes, browsers display them without any border.
- Styling a table with CSS can greatly improve its appearance.
- With CSS table properties, we can set how tables and table columns should be displayed.

List of CSS table properties

There are several CSS properties by which we can style our HTML tables :-

1) border-spacing
2) border-collapse
3) vertical-align
4) caption-side
5) empty-cells
6) table-layout

1) "border-spacing" property

- The border-spacing CSS property sets the spacing between the borders of adjacent table cells using the separated border model.
- If the collapsing border model is used, this property is ignored.
- This property is used to set the distance between the borders of adjacent cells. It does not allow negative values.

Syntax

```
element
{
   border-spacing : value;
}
```

where,
a) element = p, div, span etc.
b) value = length

here,
length

- If two values are defined, first value defines the horizontal space and second value defines the vertical spacing.
- If only one value is given then it defines the both horizontal and vertical spacing between the adjacent borders of cells.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Document</title>
        <style>
        table   {
                        border-spacing: 20px;
                }
        </style>
</head>
<body>
                <h1>CSS Table Properties</h1>
        <table border="2">
                <tr>
                        <td>Table Data 1</td>
                        <td>Table Data 2</td>
                        <td>Table Data 3</td>
                        <td>Table Data 4</td>
                </tr>
                <tr>
                        <td>Table Data 1</td>
                        <td>Table Data 2</td>
                        <td>Table Data 3</td>
                        <td>Table Data 4</td>
                </tr>
                <tr>
                        <td>Table Data 1</td>
                        <td>Table Data 2</td>
                        <td>Table Data 3</td>
                        <td>Table Data 4</td>
                </tr>
        </table>
</body>
</html>
```

2) "border-collapse" property

- The border-collapse CSS property sets whether cells inside a <table> have shared or separate borders.
- The border-collapse property in CSS is used to set the borders of the cell present inside the table and tells whether these cells will share a common border or not.

Syntax

```
element
{
   border-collapse : value;
}
```

where,
a) element = p, div, span etc.
b) value = seperate, collapse etc.

here

seperate -- This property is used to set separate border of a cell. This is default.
collapse -- It is used to collapse adjacent cells and make common border.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<meta http-equiv="X-UA-Compatible" content="ie=edge">
	<title>Document</title>
	<style>
		table	{
			border-collapse: collapse;
		}
	</style>
</head>
<body>
	<h1>CSS Table Properties</h1>
	<table border="2">
		<tr>
			<td>Table Data 1</td>
			<td>Table Data 2</td>
			<td>Table Data 3</td>
			<td>Table Data 4</td>
		</tr>
		<tr>
			<td>Table Data 1</td>
			<td>Table Data 2</td>
			<td>Table Data 3</td>
			<td>Table Data 4</td>
		</tr>
		<tr>
			<td>Table Data 1</td>
			<td>Table Data 2</td>
			<td>Table Data 3</td>
			<td>Table Data 4</td>
		</tr>
	</table>
</body>
</html>
```

3) "vertical-align" property

- The vertical-align CSS property sets vertical alignment of an inline, inline-block or table-cell box.
- The vertical-align property cannot be used for the alignment of block level elements like p, ol, ul, h1, div etc.

The vertical-align property can be used in two contexts :-

- The "vertically align" an inline element's box inside its containing line box. For example, it could be used to vertically position an <img> in a line of text.
- The "vertically align" the content of a cell in a table.

Syntax

```
element
{
    vertical-align : value;
}
```

where,
a) element = p, div, span etc.
b) value = baseline, length, sub, super, top, text-top, middle, bottom, text-bottom etc.

here

- **baseline:** It aligns the element baseline corresponding to the baseline of the parent. This might vary according to the browser. It is the default value.
- **sub:** It aligns the element baseline corresponding to the subscript-baseline of its parent.
- **super:** It aligns the element baseline corresponding to the superscript-baseline of its parent.
- **text-top:** It aligns the element top corresponding to the top of the parent's font.
- **text-bottom:** Align the element's bottom corresponding to the bottom of the parent's font.
- **middle:** Aligns the element's middle corresponding to the middle of the parent.

- **top:** Aligns the element's top corresponding to the top of the tallest element on it's line.
- **bottom:** Aligns the element's bottom corresponding to the top of the shortest element on it's line.
- **length:** Aligns the baseline of the element to the given length above the baseline of its parent. A negative value is allowed.
- **percentage:** Aligns the element's baseline corresponding to the given percentage above the baseline of its parent, with the value being a percentage of the line-height property.

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>
      Document
   </title>
   <style>
                  h2{
                        text-decoration: underline;
                  }
      .img1 {
         vertical-align: baseline;
      }
      .img2 {
         vertical-align: sub;
      }
      .img3 {
         vertical-align: super;
      }
      .img4 {
         vertical-align: text-top;
      }
      .img5 {
         vertical-align: text-bottom;
      }
      .img6 {
         vertical-align: middle;
      }
      .img7 {
         vertical-align: top;
      }
      .img8 {
         vertical-align: bottom;
      }

.img9 {
         vertical-align: 25px;
      }
```

```
      img.img10 {
         vertical-align: 100%;
      }
      .img11 {
         vertical-align: baseline;
      }
      .img12 {
         vertical-align: inherit;
      }
   </style>
</head>
 <body>
        <h1>CSS Table Properties</h1>
   <h2>vertical-align: baseline</h2>
<p>
      <img class = "img1" src ="pic.png"
      width="100" height="50" />
      Learn To Earn Labs
   </p><br>
   <h2>vertical-align: sub</h2>
<p>
      <img class = "img2" src = "pic.png"
      width="100" height="50" />
      Learn To Earn Labs
   </p><br>
   <h2>vertical-align: super</h2>
<p>
      <img class = "img3" src = "pic.png"
      width="100" height="50" />
      Learn To Earn Labs
   </p><br>
   <h2>vertical-align: text-top</h2>
<p>
      <img class = "img4" src = "pic.png"
      width="100" height="50" />
      Learn To Earn Labs
   </p><br>
   <h2>vertical-align: text-bottom</h2>     <p>
      <img class = "img5" src = "pic.png"
      width="100" height="50" />
      Learn To Earn Labs
```

```
    </p><br>
    <h2>vertical-align: middle</h2>     <p>
       <img class = "img6" src = "pic.png"
       width="100" height="50" />
       Learn To Earn Labs
    </p><br>
    <h2>vertical-align: top</h2>     <p>
       <img class = "img7" src = "pic.png"
       width="100" height="50" />
       Learn To Earn Labs
    </p><br>
    <h2>vertical-align: bottom</h2>
    <p>
       <img class = "img8" src = "pic.png"
       width="100" height="50" />
       Learn To Earn Labs
    </p><br>
    <h2>vertical-align: length</h2>
    <p>
       <img class = "img9" src = "pic.png"
       width="100" height="50" />
       Learn To Earn Labs
    </p><br>
    <h2>vertical-align: percentage</h2>
    <p>
       <img class = "img10" src = "pic.png"
       width="100" height="50" />
       Learn To Earn Labs
    </p><br>
    <h2>vertical-align: initial</h2>     <p>
       <img class = "img11" src = "pic.png"
       width="100" height="50" />
       Learn To Earn Labs
    </p><br>
    <h2>vertical-align: inherit</h2>     <p>
       <img class = "img12" src = "pic.png"
       width="100" height="50" />
       Learn To Earn Labs
    </p>
    </body>
</html>
```

4) "caption-side" property

- The caption-side CSS property puts the content of a table's <caption> on the specified side.
- This property specify's the position of table's caption.
- This property is used to specify the position where the table caption is placed.
- It is used in HTML Tables.

Syntax

```
element
{
   caption-side : value;
}
```

where,
a) element = p, div, span etc.
b) value = top, bottom etc.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Document</title>
        <style>
                table           {
                        caption-side: bottom;
                }
                </style>
</head>
<body>
                <h1>CSS Table Properties</h1>
        <table border="2">
                <caption>Table Caption</caption>
                <tr>
                        <td>Table Data 1</td>
                        <td>Table Data 2</td>
                        <td>Table Data 3</td>
                        <td>Table Data 4</td>    </tr>
                <tr>
                        <td>Table Data 1</td>
                        <td>Table Data 2</td>
                        <td>Table Data 3</td>
                        <td>Table Data 4</td>            </tr>
                                <tr>
                        <td>Table Data 1</td>
                        <td>Table Data 2</td>
                        <td>Table Data 3</td>
                        <td>Table Data 4</td>            </tr>
        </table>
</body>
</html>
```

5) "empty-cells" property

- The empty-cells CSS property show or hide borders and backgrounds of table cells that have no visible content.
- A non-breaking space ( ) is considered as a visible content.

Syntax

```
element
{
   empty-cells : value;
}
```

where,
a) element = p, div, span etc.
b) value = show, hide, initial, inherit, etc.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<meta http-equiv="X-UA-Compatible" content="ie=edge">
	<title>Document</title>
	<style>
		table		{
			empty-cells: hide;
		}
	</style>
</head>
<body>
		<h1>CSS Table Properties</h1>
	<table border="2">
		<caption>Table Caption</caption>
		<tr>
			<td>Table Data 1</td>
			<td>Table Data 2</td>
			<td>Table Data 3</td>
			<td>Table Data 4</td>		</tr>
		<tr>
			<td>Table Data 1</td>
			<td>Table Data 2</td>
			<td>Table Data 3</td>
			<td>Table Data 4</td>		</tr>
		<tr>
			<td>Table Data 1</td>
			<td>Table Data 2</td>
			<td>Table Data 3</td>
			<td></td>		</tr>
	</table>
</body>
</html>
```

6) "table-layout" property

- The table-layout CSS property sets the algorithm used to layout <table> cells, rows, and columns.
- The table-layout property in CSS is used to display the layout of table.

Syntax

```
element
{
   table-layout : value;
}
```

where,
a) element = p, div, span etc.
b) value = auto, fixed etc.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Document</title>
<style>
   table {
      width: 250px;
      border-collapse: separate;
   }
   table, tr, th, td{
      border: 1px solid #000000;
   }
   .auto {
      table-layout: auto;
   }
   .fixed {
      table-layout: fixed;
   }
   td{
      width: 50%;
   }
</style>
</head>
<body>
         <h1>CSS Table Properties</h1>
   <table class="auto">
      <caption>Example 1. Auto</caption>
      <tr>
         <th>Name</th>
         <td>Mohit Singh</td>
      </tr>
      <tr>
         <th>Email</th>
         <td>mohit9pages@gmail.com</td>
```

```
    </tr>
  </table>
  <br>
  <table class="fixed">
    <caption>Example 2. Fixed</caption>
    <tr>
      <th>Name</th>
      <td>Shubham Agrawal</td>
    </tr>
    <tr>
      <th>Email</th>
      <td>shubham1993@gmail.com</td>
    </tr>
  </table>
</body>
</html>
```

CSS3 Displays

- CSS display is the most important property of CSS which is used to control the layout of the element. It specifies how the element is displayed.
- The CSS specification defines the default display value for all the elements, e.g. the element is rendered as block, while the element is displayed inline.
- Overriding the default display value of an element is an important implication of the display property.
- The display property just changes the display behavior of an element, NOT the type of element it is.

Syntax

```
element
{
   display : value;
}
```

where,
a) element -- span, div etc.
b) value -- none, inline, block, inline-block, list-items, flex, inline-flex, grid, inline-grid, table,inline-table, table-row, table-cell, table-column, table-caption, table-caption-group, table-row-group, table-header-group, table-footer-group, run-in, contents etc.

Example -- inline

- The inline value of the display property causes an element to behave as though it were an inline-level element, like a or an <a> element.
- Inline property doesn't allows width and height for their elements.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>Example of CSS display</title>
 <style>
        p {
            display: inline;
            background: red;
            padding: 10px;
        }
        ul li {
            display: inline;
            margin: 10px;
        }
 </style>
</head>
<body>
 <p>This paragraph element generates an inline box.</p>
        <ul>
                <li>Item 1</li>
                <li>Item 2</li>
                <li>Item 3</li>
                <li>Item 4</li>
                <li>Item 5</li>
        </ul>
</body>
</html>
```

Example -- block

The block value of the display property forces an element to behave like block-level element, like a or <p> element.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>Document</title>
   <style>
            a      {
               display: block;
               color: green;
               padding: 3px;
            }
            span      {
               display: block;
               background-color: red;
               color: white;
               padding: 3px;
            }
   </style>
</head>
<body>
   <a href="#hello">Anchor Tag [An Inline Element]</a>
<span>
   Span Tag --
   Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatem ut error tenetur non dolor vel omnis accusantium totam commodi, dignissimos, repellat molestiae. Cum, voluptatem deleniti natus aperiam quos quibusdam! Dignissimos?
</span>
</body></html>
```

Example -- inline-block

The CSS display inline-block element is very similar to inline element but the difference is that we are able to set the width and height.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
            a       {
                display: inline-block;
                color: green;
                width: 800px;
                height: 100px;
            }
            span       {
                display: inline-block;
                background-color: red;
                color: white;
                padding: 3px;
            }
    </style>
</head>
<body>
    <a href="#hello">Anchor Tag [An Inline-block Element]</a>
<span>
    Span Tag --
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatem ut error tenetur non dolor vel omnis accusantium totam commodi, dignissimos, repellat molestiae. Cum, voluptatem deleniti natus aperiam quos quibusdam! Dignissimos?
</span>
</body></html>
```

Example -- none

The value none for the display property does not create an invisible box — it creates no box at all.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>Document</title>
   <style>
      a       {
         display: none;
         color: green;
         width: 800px;
         height: 100px;
      }
      span       {
         display: inline-block;
         background-color: red;
         color: white;
         padding: 3px;
      }
   </style>
</head>
<body>
   <a href="#hello">Anchor Tag [An Inline Element]</a>
<span>
   Span Tag --
   Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatem ut error tenetur non dolor vel omnis accusantium totam commodi, dignissimos, repellat molestiae. Cum, voluptatem deleniti natus aperiam quos quibusdam! Dignissimos?
</span>
</body></html>
```

Example -- list-item

- The "list-item" is similar to block as it takes up the whole width available, but shows an additional bullet point.
- A single value of list-item will cause the element to behave like a list item.
- When we apply display:list-item then we can use all types of list property on that element like list-style-type and list-style-position properties.

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<meta http-equiv="X-UA-Compatible" content="ie=edge">
	<title>Document</title>
	<style>
		span
		{
			display: list-item;
			list-style-position: inside;
		}

	</style>
</head>
<body>

	<span>Hello Span Tag 1</span>
	<span>Hello Span Tag 2</span>
	<span>Hello Span Tag 3</span>
	<span>Hello Span Tag 4</span>
	<span>Hello Span Tag 5</span>

</body>
</html>
```

Example -- run-in

- In CSS3, a new display value was introduced: run-in.
- The "run-in" does is it allows us to have block-level elements behave like inline elements. We can have a block-level element behave like an inline element by setting its display to inline, of course, but there are certain situations where that would put the semantics of the elements at risk.
- The element generates a run-in box. If the adjacent sibling of the element defined as display: run-in box is a block box, the run-in box becomes the first inline box of the block box that follows it.
- The "run-in" property is experimental and currently not in use, hence it is recommended to not use this property.
- The "run-in" property is not supported in firefox and older versions of Google & Opera.

Example -- contents

- The "display: contents" causes an element's children to appear as if they were direct children of the element's parent, ignoring the element itself.
- This can be useful when a wrapper element should be ignored when using CSS grid or similar layout techniques.
- The "contents" property is experimental and currently not in use, hence it is recommended to not use this property.
- The "contents" property is only supported in firefox and not supported by other browsers.

Visibility in CSS3

- The CSS visibility property is used to specify whether an element is visible or not.
- By using CSS visibility property we can makes an element invisible but an invisible element also take up the space on the page.
- By using display property you can create invisible elements that don't take up space.

Syntax

```
element
{
   visibility : value;
}
```

where,
a) element = div, p, span etc.
b) value = visible, hidden, collapse.

Example -- visible

- It is the default value. The element is show or visible normally in the web document.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        div       {
            border: 5px solid black;
            padding: 5px;
            width: 600px;
            visibility: visible;
      }
      </style>
</head>
<body>

    <h1>CSS3 Visibility Property</h1>
    <div>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Iure cum eaque in illo eum ipsam consequuntur ut, nulla sint, doloribus esse error sed distinctio minus hic. Similique debitis iste rerum nisi optio numquam velit voluptate, earum a. Iure minus ad necessitatibus labore, perferendis illum ipsum quos dicta laudantium ex quidem rerum possimus sint? Asperiores delectus tempore odio iste doloribus, reprehenderit dolore cupiditate neque voluptates, quos ratione dolores officia mollitia ad provident, eos error porro eveniet perferendis totam in inventore animi deserunt vel. Id voluptates quidem doloribus? Voluptates rem inventore, voluptate, cum nihil ea distinctio eos a, in amet minima. Voluptas!</div>
</body>
</html>
```

Example -- hidden

- This property hide the element from the page but takes up space in the document.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        div       {
            border: 5px solid black;
            padding: 5px;
            width: 600px;
            visibility: hidden;
      }
        </style>
</head>
<body>
        <h1>CSS3 Visibility Property</h1>
    <div>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Iure cum eaque in illo eum ipsam consequuntur ut, nulla sint, doloribus esse error sed distinctio minus hic. Similique debitis iste rerum nisi optio numquam velit voluptate, earum a. Iure minus ad necessitatibus labore, perferendis illum ipsum quos dicta laudantium ex quidem rerum possimus sint? Asperiores delectus tempore odio iste doloribus, reprehenderit dolore cupiditate neque voluptates, quos ratione dolores officia mollitia ad provident, eos error porro eveniet perferendis totam in inventore animi deserunt vel. Id voluptates quidem doloribus? Voluptates rem inventore, voluptate, cum nihil ea distinctio eos a, in amet minima. Voluptas!</div>
    <p>Hello This is paragraph without any margin but because the element is just hide with the visibility property the paragraph content shows here.</p>
</body>
</html>
```

Example -- collapse

- This property only used for the table elements. It is used to remove the rows and column from the table but it does not affect the layout of the Table. But their space is available for other content.

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      CSS | visibility Property
    </title>
    <style>
      table.hello {
        visibility: collapse
      }
      table, th, td {
      border:1px solid red;
      }
      p {
      color:green;
      font-size:25px;
      }
    </style>
  </head>
  <body>
    <h1 style="color:green;">CSS3 Visibility Property</h1>
    <table style="border:1px solid red;" class="hello">
      <tr>
        <th>HTML</th>
        <th>CSS</th>
        <th>JavaScript</th>
      </tr>
    </table>
     </body>
</html>
```

Display Vs Visibility Property

Display

- The "display: none;" turns off the display and removes the element completely from the document.
- It does not take up any space, even though the HTML for it is still in the source code.
- All child elements also have their display turned off, even if their display property is set to something other than none.

Visibility

- The "visibility: hidden;" hides the element, but it still takes up space in the layout.
- Child element of a hidden box will be visible if their visibility is set to visible.

CSS3 Positioning

- The CSS position property is used to set position for an element.
- The position property defines how an element will be positioned on a page.
- Positioning elements appropriately on the web pages is a necessity for a good layout design.
- The positioning of an element can be done using the top, right, bottom and left property.

Syntax

```
element
{
   position : value;
}
```

where,
a) element = div, p, span etc.
b) value = static, fixed, relative, absolute, sticky

Types of CSS Positioning Methods

There are five different types of position property available in CSS :

a) Absolute Positioning

- An absolutely positioned element is positioned relative to the first parent element that has a position other than static.
- If no such element is found, it will be positioned on a page relative to the 'top-left' corner of the browser window.

Or,

- If no element is found, the containing block(parent) is HTML.
- With the absolute positioning, we can place an element anywhere on a page.
- We can set the position of absolute positioned element using the top, right, bottom, left.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        .relative
        {
            width: 100%;
            height: 200px;
            font-size: 20px;
            background-color: lime;
            text-align: center;
            line-height: 70px;
            position: relative;
        }
          .absolute
        {
            width: 500px;
            height: 100px;
            font-size: 20px;
            background-color: blue;
            text-align: center;
            bottom: 20px;
            color: white;
            left: 20px;
            position: absolute;
        }
        </style>
</head>
<body>

    <h1>CSS Position Property</h1>
```

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Voluptates ea qui quasi similique unde debitis autem nam modi provident culpa reiciendis natus, incidunt adipisci recusandae iusto blanditiis? Illo, quisquam omnis! Asperiores molestiae non, tempore doloremque ex quas laboriosam quaerat a cum culpa corrupti libero et magni repellat ipsum expedita similique reprehenderit facere esse harum eius delectus omnis commodi sint? Sequi nam quisquam et natus aperiam vel rerum, ratione, porro ipsam pariatur obcaecati laudantium voluptatum officiis nesciunt, neque eaque quam provident illum ipsum amet? Dolorum blanditiis quos, ipsa maiores id quas repellendus, illum.

Relative Positioning
Absolute Positioning -- Move relative to Parent

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestias at similique, exercitationem aspernatur deserunt nihil? Cupiditate animi maiores magnam inventore, reprehenderit, recusandae perspiciatis laborum ipsam alias repellendus praesentium eius eaque dignissimos aut facilis eos reiciendis accusantium aliquam cum. Debitis hic quaerat quia provident placeat tenetur, suscipit asperiores, vero, necessitatibus numquam fuga obcaecati pariatur quos totam ullam. Maxime excepturi fugiat perspiciatis, quidem mollitia illo natus nam doloremque repellat atque, rerum voluptatem error a distinctio nesciunt sed laboriosam. aut non molestias animi nostrum ta! Sed doloremque distinctio aperiam, doloribus nihil, dolorum consequatur cupiditate voluptatem, pariatur dolor consectetur cumque velit. Non qui ipsa, animi aspernatur veritatis iure, tempora repellat odio nobis aperiam minima quo saepe natus eaque vero. Eos.

</body>
</html>

b) Static Positioning

- This is a by default position for HTML elements. It always positions an element according to the normal flow of the page.
- It is not affected by the top, bottom, left and right properties.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        .div1
        {
            width: 400px;
            height: 100px;
            font-size: 20px;
            background-color: lime;
            text-align: center;
            line-height: 70px;
            position: static;
        }
        .div2
        {
            width: 400px;
            height: 100px;
            font-size: 20px;
            background-color: blue;
            text-align: center;
            line-height: 70px;
            position: static;
            color: white;
        }
```

```
        .div3
        {
            width: 400px;
            height: 100px;
            font-size: 20px;
            background-color: red;
            text-align: center;
            line-height: 70px;
            position: static;
            color: white;
        }

    </style>
</head>
<body>

    <h1>CSS Position Property</h1>

    <div class="div1">
        Static Position
    </div>

    <div class="div2">
        Static Position
    </div>

    <div class="div3">
        Static Position
    </div>

</body>
</html>
```

c) Relative Positioning

- A relative positioned element is positioned relative to its normal position.
- An element with "position: relative" is positioned relatively with the other elements which are sitting at top of it.
- If we set its top, right, bottom or left, other elements will not fill up the gap left by this element.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>

        .div2
        {
            width: 100%;
            height: 100px;
            font-size: 20px;
            background-color: blue;
            text-align: center;
            line-height: 70px;
            position: relative;
            color: white;
        }

    </style>
</head>
<body>

    <h1>CSS Position Property</h1>
```

```
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ipsum, qui quibusdam sapiente ex facilis sit, incidunt officia molestiae neque consectetur, dolor cum deserunt reprehenderit! Ipsa, possimus error explicabo dignissimos ut perferendis, quisquam necessitatibus labore est inventore vitae amet dolores exercitationem, porro cumque vel natus voluptatem! Adipisci mollitia dolorum a natus quis cumque minima quisquam! Repudiandae quos asperiores aspernatur earum qui vel omnis pariatur dicta provident, iure dolores tenetur eos, ea maiores voluptatem doloremque? Rerum praesentium delectus asperiores ipsum ut aspernatur nihil! Reprehenderit enim placeat eligendi repudiandae modi libero pariatur perspiciatis quidem sequi excepturi voluptate, magnam repellat ipsa possimus? Eligendi aliquid cupiditate sed, exercitationem qui quis unde voluptatum dignissimos ad nihil, natus vitae nisi ea, rerum praesentium fuga?</p>

    <div class="div2">
       Relative Position
    </div>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quidem consectetur, obcaecati facilis officia odio dolorum fuga praesentium reprehenderit, porro atque accusamus necessitatibus cumque eaque omnis sequi tempora natus dolores nesciunt perferendis debitis voluptate rerum suscipit deleniti? Ducimus, ipsam numquam? Iste porro? Eius aspernatur, molestias eveniet atque ipsa molestiae deserunt fuga, dicta explicabo obcaecati, doloribus, molestias debitis labore quidem blanditiis nobis c ipsum, accusamus minima officia sequi est illo! Optio sit quibusdam fuga nostrum sed quaerat consectetur tempora facere molestiae eveniet harum in dolor doloribus, officia magni fugiat soluta aspernatur excepturi porro ipsa incidunt, doloremque, labore veniam? Corporis voluptatibus, totam quaerat optio, ex deleniti architecto asperiores itaque numquam dignissimos nostrum? Saepe assumenda voluptatum eveniet in incidunt obcaecati qui unde ullam. Inventore iusto molestias consequuntur soluta consequatur tenetur, corporis itaque minus suscipit perferendis voluptatem! Architecto, nobis.</p>

</body>
</html>
```

d) Fixed Positioning

- Fixed positioning is a subcategory of absolute positioning.
- The only difference is, a fixed positioned element is fixed with respect to the browser's viewport and does not move when scrolled.
- We can set the position of fixed positioned element using the top, right, bottom, left.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>

        .div2
        {
            width: 100%;
            height: 100px;
            font-size: 20px;
            background-color: blue;
            text-align: center;
            line-height: 70px;
            position: fixed;
            color: white;
        }

    </style>
</head>
<body>

    <h1>CSS Position Property</h1>
```

```
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ipsum, qui quibusdam sapiente ex facilis sit, incidunt officia molestiae neque consectetur, dolor cum deserunt reprehenderit! Ipsa, possimus error explicabo dignissimos ut perferendis, quisquam necessitatibus labore est inventore vitae amet dolores exercitationem, porro cumque vel natus voluptatem! Adipisci mollitia dolorum a natus quis cumque minima quisquam! Repudiandae quos asperiores aspernatur earum qui vel omnis pariatur dicta provident, iure dolores tenetur eos, ea maiores voluptatem doloremque? Rerum praesentium dolorum, repellendus excepturi atque laudantium, dignissimos sapiente delectus placeat accusantium facere voluptatibus incidunt distinctio consequuntur earum natus? Fuga animi, officia magnam repellat ipsa possimus? Eligendi aliquid cupiditate sed, exercitationem qui quis unde voluptatum dignissimos ad nihil, natus vitae nisi ea, rerum praesentium fuga?</p>

    <div class="div2">
      Fixed Position
    </div>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quidem consectetur, obcaecati facilis officia odio dolorum fuga praesentium reprehenderit, porro atque accusamus necessitatibus cumque eaque omnis sequi tempora natus dolores nesciunt atque eos fuga quo doloremque, officiis inventore veniam dolorem saepe asperiores. Repellat est totam quisquam. Consequatur vero laborum repellat similique iusto nulla voluptatem minus ex beatae sapiente deserunt dicta. Vitae, ad sit voluptates ducimus quis nam ratione nulla id totam perspiciatis vero at fugit nihil exercitationem doloremque quae! Dignissimos non eaque, nemo ipsum est aut assumenda ex quas placeat rem. Voluptatibus, tenetur quidem id enim error soluta aliquam aspernatur aut. Deserunt rem, voluptatum nemo possimus tenetur ullam consequatur perspiciatis cupiditate fugiat quibusdam, voluptate doloremque amet nihil ea quae? Sed ab officiis eaque, distinctio quibusdam fugiat consequuntur ex? Exercitationem corporis assumenda, ipsam voluptatibus tempora consequuntur, expedita et quisquam rem voluptas magnam.</p>

</body>
</html>
```

e) Sticky Positioning

- Sticky positioning is a hybrid of relative and fixed positioning.
- A sticky element toggles between relative and fixed , depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>

        .div2
        {
            width: 100%;
            height: 100px;
            font-size: 20px;
            background-color: blue;
            text-align: center;
            line-height: 70px;
            top: 10px;
            position: sticky;
            color: white;
        }

    </style>
</head>
<body>

    <h1>CSS Position Property</h1>
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Ipsum, qui quibusdam sapiente ex facilis sit, incidunt officia molestiae neque consectetur, dolor cum deserunt reprehenderit! Ipsa, possimus error explicabo dignissimos ut perferendis, quisquam aut voluptate eius, amet iste deserunt aliquid totam, repellat eveniet deleniti nemo ipsam distinctio quae, similique soluta alias cumque. Minus aperiam corrupti, impedit ex atque modi consequuntur veritatis quibusdam animi numquam unde eligendi voluptas quidem nostrum, facere, dignissimos quis? Odit totam quia neque corrupti ipsam dicta laudantium, laboriosam inventore ducimus animi sequi dolorum adipisci facilis consectetur. Pariatur dolorum, repellendus excepturi atque laudantium, dignissimos sapiente delectus placeat accusantium facere voluptatibus incidunt distinctio consequuntur earum natus? Fuga animi, officia magnam repellat ipsa possimus? Eligendi aliquid cupiditate sed, exercitationem qui quis unde voluptatum dignissimos ad nihil, natus vitae nisi ea, rerum praesentium fuga?

Relative Position

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quidem consectetur, obcaecati facilis officia odio dolorum fuga praesentium reprehenderit, porro atque accusamus necessitatibus cumque eaque omnis sequi tempora natus dolores nesciunt perferendis debitis voluptate rerum suscipit deleniti? Ducimus, ipsam numquam? Iste architecto asperiores fugit illum dolorum reiciendis alias quod incidunt, consequuntur dolores dolorem, id, maxime hic vitae facere distinctio veritatis? In, voluptates, alias suscipit officiis aspernatur ducimus dolorum a voluptatibus itaque accusamus, fugiat aut blanditiis similique aliquam dicta sit pariatur dolore. Consectetur, dolorum?

</body>
</html>

CSS3 Layering

- The z-index property is used to displace elements on the z-axis i.e in or out of the screen.
- It is used to define the order of elements if they overlap on each other.
- The CSS z-index property can be used in conjugation with the position property to create an effect of layers like Photoshop.
- Overlapping elements with a larger z-index cover those with a smaller one.
- By default the z-index of every element is 0 if they are absolute, fixed, sticky or relative positioned.
- Elements with non-static positioning will always appear on top of elements with default static positioning.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        .div1,.div2,.div3,.div4,.div5,.div6,.div7,.div8      {
            width: 100px;
            height: 100px;
            position: fixed;
        }
        .div1       {
            background-color: red;
        }
        .div2       {
            background-color: blue;
            margin-top: 20px;
            margin-left: 20px;
        }
```

```
    .div3       {
            background-color: green;
            margin-top: 40px;
            margin-left: 40px;
        }
        .div4       {
            background-color: yellow;
            margin-top: 60px;
            margin-left: 60px;
        }
        .div5       {
            background-color: red;
            margin-top: 240px;
            margin-left: 20px;
            z-index: 2;
        }
        .div6       {
            background-color: blue;
            margin-top: 260px;
            margin-left: 40px;
        }
        .div7       {
            background-color: green;
            margin-top: 280px;
            margin-left: 60px;
            z-index: 1;
        }
        .div8       {
            background-color: yellow;
            margin-top: 300px;
            margin-left: 80px;
        }

    </style>
</head>
```

```
<body>

    <h1>Before Z-Index</h1>

    <div class="div1"></div>
    <div class="div2"></div>
    <div class="div3"></div>
    <div class="div4"></div>

    <h1 style="position: absolute;top: 240px;">After Z-Index -- We can change the stack order</h1>

    <div class="div5"></div>
    <div class="div6"></div>
    <div class="div7"></div>
    <div class="div8"></div>

</body>
</html>
```