FLEXBOX

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

Flexbox (flexible box) is a layout mode of CSS3. Using this mode, you can easily create layouts for complex applications and web pages. Flexbox layout gives complete control over the direction, alignment, order, size of the boxes.

In this tutorial, we are going to learn how to use the various features available in Flexbox.

Audience

This tutorial has been prepared for beginners to make them understand the basics of Flexbox library. It will help the readers in aligning the contents of a webpage easily.

Prerequisites

For this tutorial, it is assumed that the readers have a prior knowledge of CSS and HTML

Copyright & Disclaimer

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

1. FLEXBOX – OVERVIEW

Cascading **S**tyle **S**heets (CSS) is a simple design language intended to simplify the process of making web pages presentable. CSS handles the look and feel part of a web page.

Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

To determine the position and dimensions of the boxes, in CSS, you can use one of the layout modes available –

- **The block layout**: This mode is used in laying out documents.
- **The inline layout**: This mode is used in laying out text.
- **The table layout**: This mode is used in laying out tables.
- **The positioned layout**: This mode is used in positioning the elements.

All these modes are used to align specific elements like documents, text, tables, etc., however, none of these provides a complete solution to lay out complex websites. Initially this is used to be done using a combination of floated elements, positioned elements, and table layout (often). But floats only allow to horizontally position the boxes.

What is Flexbox?

In addition to the above-mentioned modes, CSS3 provides another layout mode Flexible Box, commonly called as **Flexbox**.

Using this mode, you can easily create layouts for complex applications and web pages. Unlike floats, Flexbox layout gives complete control over the direction, alignment, order, size of the boxes.

Features of Flexbox

Following are the notable features of Flexbox layout:

- **Direction:** You can arrange the items on a web page in any direction such as left to right, right to left, top to bottom, and bottom to top.

- **Order:** Using Flexbox, you can rearrange the order of the contents of a web page.

- **Wrap:** In case of inconsistent space for the contents of a web page (in single line), you can wrap them to multiple lines (both horizontally) and vertically.

- **Alignment:** Using Flexbox, you can align the contents of the webpage with respect to their container.

- **Resize**: Using Flexbox, you can increase or decrease the size of the items in the page to fit in available space.

Supporting browsers

Following are the browsers that support Flexbox.

- Chrome 29+
- Firefox 28+
- Internet Explorer 11+
- Opera 17+
- Safari 6.1+
- Android 4.4+
- iOS 7.1+

2. FLEXBOX – FLEX CONTAINERS

To use Flexbox in your application, you need to create/define a flex container using the **display** property.

Usage:

```
display: flex | inline-flex
```

This property accepts two values –

- **flex:** Generates a block level flex container.
- **inline-flex:** Generates an inline flex container box.

Now, we will see how to use the **display** property with examples.

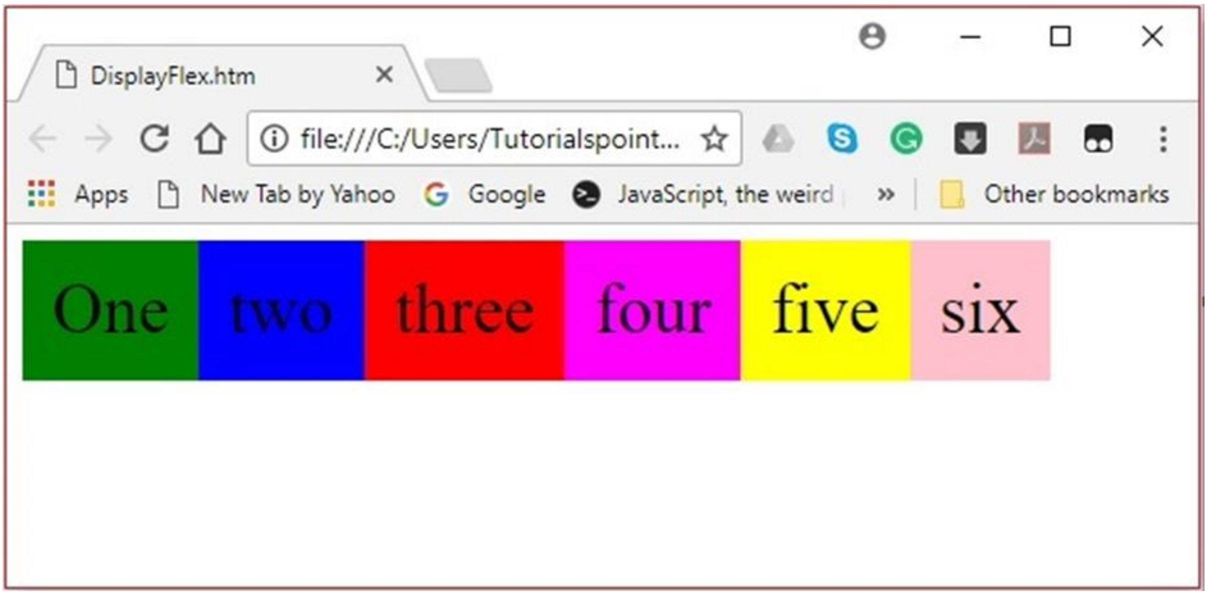
Flex

On passing this value to the display property, a block level flex container will be created. It occupies the full width of the parent container (browser).

The following example demonstrates how to create a block level flex container. Here, we are creating six boxes with different colors and we have used the flex container to hold them.

```
<!doctype html>
<html lang="en">

<style>
.box1{background:green;}
.box2{background:blue;}
.box3{background:red;}
.box4{background:magenta;}
.box5{background:yellow;}
.box6{background:pink;}

.container{
   display:flex;
}

.box{
  font-size:35px;
  padding:15px;
}
</style>

<body>
<div class="container">
  <div class="box box1">One</div>
  <div class="box box2">two</div>
  <div class="box box3">three</div>
  <div class="box box4">four</div>
  <div class="box box5">five</div>
  <div class="box box6">six</div>
</div>
</body>
<html>
```

It will produce the following result –



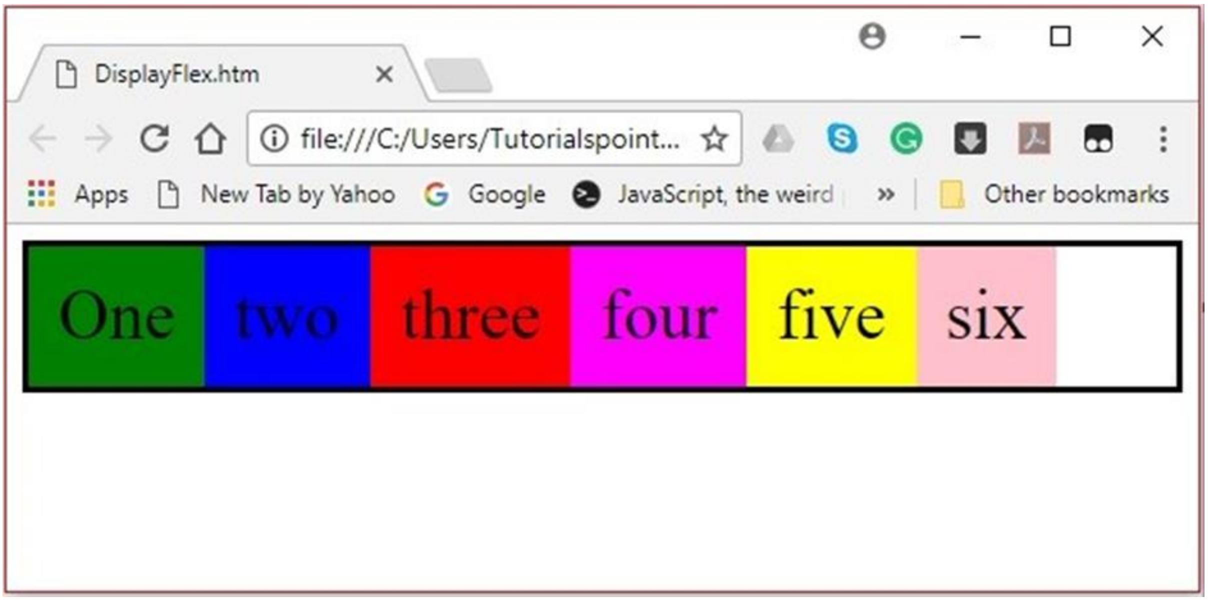
One two three four five six

Since we have given the value **flex** to the **display** property, the container uses the width of the container (browser).

You can observe this by adding a border to the container as shown below.

```
.container{
   display:inline-flex;
   border:3px solid black;
}
```

It will produce the following result –



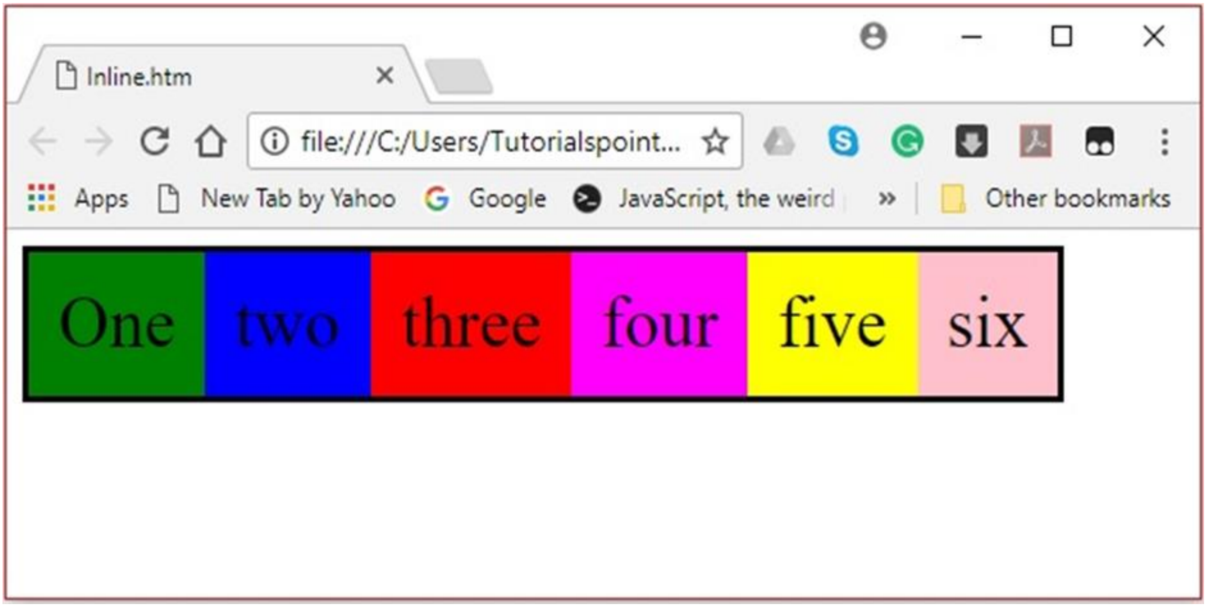
Inline flex

On passing this value to the **display** property, an inline level flex container will be created. It just takes the place required for the content.

The following example demonstrates how to create an inline flex container. Here, we are creating six boxes with different colors and we have used the inline-flex container to hold them.

```
<!doctype html>
<html lang="en">
<style>
.box1{background:green;}
.box2{background:blue;}
.box3{background:red;}
.box4{background:magenta;}
.box5{background:yellow;}
.box6{background:pink;}

.container{
   display:inline-flex;
   border:3px solid black;
}

.box{
  font-size:35px;
  padding:15px;
}

</style>
<body>
<div class="container">
  <div class="box box1">One</div>
  <div class="box box2">two</div>
  <div class="box box3">three</div>
  <div class="box box4">four</div>
  <div class="box box5">five</div>
  <div class="box box6">six</div>
</div>
</body>
<html>
```

It will produce the following result –



Since we have used an inline flex container, it just took the space that is required to wrap its elements.