

CSS3 Flexible Box

[< Previous](#)[Next >](#)

CSS3 Flexbox

Flexible boxes, or flexbox, is a new layout mode in CSS3.

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

For many applications, the flexible box model provides an improvement over the block model in that it does not use floats, nor do the flex container's margins collapse with the margins of its contents.

Browser Support

The numbers in the table specify the first browser version that fully supports the feature.

Numbers followed by -webkit- or -moz- specify the first version that worked with a prefix.

Property					
Basic support (single-line flexbox)	29.0 21.0 - webkit-	11.0	22.0 18.0 -moz-	6.1 -webkit-	12.1 - webkit-
Multi-line flexbox	29.0 21.0 - webkit-	11.0	28.0	6.1 -webkit-	17.0 15.0 - webkit- 12.1

CSS3 Flexbox Concepts

Flexbox consists of flex containers and flex items.

[HTML](#) [CSS](#) [MORE ▼](#)

Inside a flex container there is one or more flex items.

Note: Everything outside a flex container and inside a flex item is rendered as usual. Flexbox defines how flex items are laid out inside a flex container.

Flex items are positioned inside a flex container along a flex line. By default there is only one flex line per flex container.

The following example shows three flex items. They are positioned by default: along the horizontal flex line, from left to right:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}

.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}
</style>
</head>
<body>

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

</body>
</html>
```

[Try it Yourself »](#)

[HTML](#) [CSS](#) [MORE ▼](#)

If we set the `direction` property to `rtl` (right-to-left), the text is drawn right to left, and also the flex line changes direction, which will change the page layout:

Example

```
body {  
    direction: rtl;  
}  
  
.flex-container {  
    display: -webkit-flex;  
    display: flex;  
    width: 400px;  
    height: 250px;  
    background-color: lightgrey;  
}  
  
.flex-item {  
    background-color: cornflowerblue;  
    width: 100px;  
    height: 100px;  
    margin: 10px;  
}
```

[Try it Yourself »](#)

Flex Direction

The `flex-direction` property specifies the direction of the flexible items inside the flex container. The default value of `flex-direction` is `row` (left-to-right, top-to-bottom).

The other values are as follows:

- `row-reverse` - If the writing-mode (direction) is left to right, the flex items will be laid out right to left
- `column` - If the writing system is horizontal, the flex items will be laid out vertically
- `column-reverse` - Same as column, but reversed

The following example shows the result of using the `row-reverse` value:

[HTML](#) [CSS](#) [MORE ▼](#)

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-direction: row-reverse;  
  flex-direction: row-reverse;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[Try it Yourself »](#)

The following example shows the result of using the `column` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-direction: column;  
  flex-direction: column;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[Try it Yourself »](#)

The following example shows the result of using the `column-reverse` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-direction: column-reverse;  
  flex-direction: column-reverse;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The justify-content Property

The **justify-content** property horizontally aligns the flexible container's items when the items do not use all available space on the main-axis.

The possible values are as follows:

- **flex-start** - Default value. Items are positioned at the beginning of the container
- **flex-end** - Items are positioned at the end of the container
- **center** - Items are positioned at the center of the container
- **space-between** - Items are positioned with space between the lines
- **space-around** - Items are positioned with space before, between, and after the lines

The following example shows the result of using the **flex-end** value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-justify-content: flex-end;  
  justify-content: flex-end;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

Try it Yourself »

The following example shows the result of using the **center** value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-justify-content: center;  
  justify-content: center;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[HTML](#) [CSS](#) [MORE ▼](#)

The following example shows the result of using the `space-between` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-justify-content: space-between;  
  justify-content: space-between;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[Try it Yourself »](#)

The following example shows the result of using the `space-around` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-justify-content: space-around;  
  justify-content: space-around;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[Try it Yourself »](#)

The align-items Property

The `align-items` property vertically aligns the flexible container's items when the items do not use all available space on the cross-axis.

The possible values are as follows:

- `stretch` - Default value. Items are stretched to fit the container
- `flex-start` - Items are positioned at the top of the container

[HTML](#) [CSS](#) [MORE ▼](#)

- **baseline** - Items are positioned at the baseline of the container

The following example shows the result of using the **stretch** value (this is the default value):

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-align-items: stretch;  
  align-items: stretch;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[Try it Yourself »](#)

The following example shows the result of using the **flex-start** value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-align-items: flex-start;  
  align-items: flex-start;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[Try it Yourself »](#)

The following example shows the result of using the **flex-end** value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-align-items: flex-end;  
  align-items: flex-end;  
}
```

[HTML](#) [CSS](#) [MORE ▼](#)

```
background-color: lightgrey;
}
```

[Try it Yourself »](#)

The following example shows the result of using the `center` value:

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-align-items: center;
  align-items: center;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}
```

[Try it Yourself »](#)

The following example shows the result of using the `baseline` value:

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-align-items: baseline;
  align-items: baseline;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}
```

[Try it Yourself »](#)

The flex-wrap Property

The `flex-wrap` property specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line.

[HTML](#) [CSS](#) [MORE ▼](#)

- **nowrap** - Default value. The flexible items will not wrap
- **wrap** - The flexible items will wrap if necessary
- **wrap-reverse** - The flexible items will wrap, if necessary, in reverse order

The following example shows the result of using the **nowrap** value (this is the default value):

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-wrap: nowrap;  
  flex-wrap: nowrap;  
  width: 300px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[Try it Yourself »](#)

The following example shows the result of using the **wrap** value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-wrap: wrap;  
  flex-wrap: wrap;  
  width: 300px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

[Try it Yourself »](#)

The following example shows the result of using the **wrap-reverse** value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;
```

[HTML](#) [CSS](#) [MORE ▼](#)

```
width: 300px;
height: 250px;
background-color: lightgrey;
}
```

[Try it Yourself »](#)

The align-content Property

The `align-content` property modifies the behavior of the `flex-wrap` property. It is similar to `align-items`, but instead of aligning flex items, it aligns flex lines.

The possible values are as follows:

- `stretch` - Default value. Lines stretch to take up the remaining space
- `flex-start` - Lines are packed toward the start of the flex container
- `flex-end` - Lines are packed toward the end of the flex container
- `center` - Lines are packed toward the center of the flex container
- `space-between` - Lines are evenly distributed in the flex container
- `space-around` - Lines are evenly distributed in the flex container, with half-size spaces on either end

The following example shows the result of using the `center` value:

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-wrap: wrap;
  flex-wrap: wrap;
  -webkit-align-content: center;
  align-content: center;
  width: 300px;
  height: 300px;
  background-color: lightgrey;
}
```

[Try it Yourself »](#)

Flex Item Properties

[HTML](#) [CSS](#) [MORE ▼](#)

The **order** property specifies the order of a flexible item relative to the rest of the flexible items inside the same container:

Example

```
.flex-item {  
    background-color: cornflowerblue;  
    width: 100px;  
    height: 100px;  
    margin: 10px;  
}  
  
.first {  
    -webkit-order: -1;  
    order: -1;  
}
```

[Try it Yourself »](#)

Margin

Setting **margin: auto;** will absorb extra space. It can be used to push flex items into different positions.

In the following example we set **margin-right: auto;** on the first flex item. This will cause all the extra space to be absorbed to the right of that element:

Example

```
.flex-item {  
    background-color: cornflowerblue;  
    width: 75px;  
    height: 75px;  
    margin: 10px;  
}  
  
.flex-item:first-child {  
    margin-right: auto;  
}
```

[Try it Yourself »](#)

Perfect Centering

[HTML](#) [CSS](#) [MORE ▼](#)

It is very easy with flexbox. Setting `margin: auto;` will make the item perfectly centered in both axis:

Example

```
.flex-item {  
    background-color: cornflowerblue;  
    width: 75px;  
    height: 75px;  
    margin: auto;  
}
```

[Try it Yourself »](#)

align-self

The `align-self` property of flex items overrides the flex container's `align-items` property for that item. It has the same possible values as the `align-items` property.

The following example sets different align-self values to each flex item:

Example

```
.flex-item {  
    background-color: cornflowerblue;  
    width: 60px;  
    min-height: 100px;  
    margin: 10px;  
}  
  
.item1 {  
    -webkit-align-self: flex-start;  
    align-self: flex-start;  
}  
  
.item2 {  
    -webkit-align-self: flex-end;  
    align-self: flex-end;  
}  
  
.item3 {  
    -webkit-align-self: center;  
    align-self: center;  
}  
  
.item4 {
```

[HTML](#) [CSS](#) [MORE ▼](#)

```
}  
  
.item5 {  
  -webkit-align-self: stretch;  
  align-self: stretch;  
}
```

[Try it Yourself »](#)

flex

The **flex** property specifies the length of the flex item, relative to the rest of the flex items inside the same container.

In the following example, the first flex item will consume 2/4 of the free space, and the other two flex items will consume 1/4 of the free space each:

Example

```
.flex-item {  
  background-color: cornflowerblue;  
  margin: 10px;  
}  
  
.item1 {  
  -webkit-flex: 2;  
  flex: 2;  
}  
  
.item2 {  
  -webkit-flex: 1;  
  flex: 1;  
}  
  
.item3 {  
  -webkit-flex: 1;  
  flex: 1;  
}
```


[Try it Yourself »](#)

More Examples

Flex Properties

Lists the CSS properties used with flexbox:



	Description 
	Specifies the type of box used for an HTML element
	Specifies the direction of the flexible items inside a flex container
	Horizontally aligns the flex items when the items do not use all available space on the main-axis
	Vertically aligns the flex items when the items do not use all available space on the cross-axis
	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
	A shorthand property for flex-direction and flex-wrap
<u>order</u>	Specifies the order of a flexible item relative to the rest of the flex items inside the same container
<u>align-self</u>	Used on flex items. Overrides the container's align-items property
<u>flex</u>	Specifies the length of a flex item, relative to the rest of the flex items inside the same container

[< Previous](#)[Next >](#)

[HTML](#) [CSS](#) [MORE ▼](#)

COLOR PICKER



LEARN MORE

[Tabs](#)
[Dropdowns](#)
[Accordions](#)
[Convert Weights](#)
[Animated Buttons](#)
[Side Navigation](#)
[Top Navigation](#)
[JS Animations](#)
[Modal Boxes](#)
[Progress Bars](#)
[Parallax](#)
[Login Form](#)
[HTML Includes](#)
[Google Maps](#)
[Loaders](#)

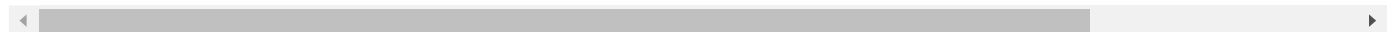
[HTML](#) [CSS](#) [MORE ▼](#)[Filter List](#)[Sort List](#)

SHARE



CERTIFICATES

HTML, CSS, JavaScript, PHP, jQuery, Bootstrap and XML.

[Read More »](#)[REPORT ERROR](#)[PRINT PAGE](#)[FORUM](#)[ABOUT](#)

[HTML](#) [CSS](#) [MORE ▼](#)

Top 10 Tutorials

- [HTML Tutorial](#)
- [CSS Tutorial](#)
- [JavaScript Tutorial](#)
- [W3.CSS Tutorial](#)
- [Bootstrap Tutorial](#)
- [SQL Tutorial](#)
- [PHP Tutorial](#)
- [jQuery Tutorial](#)
- [Angular Tutorial](#)
- [XML Tutorial](#)

Top 10 References

- [HTML Reference](#)
- [CSS Reference](#)
- [JavaScript Reference](#)
- [W3.CSS Reference](#)
- [Browser Statistics](#)
- [PHP Reference](#)
- [HTML Colors](#)
- [HTML Character Sets](#)
- [jQuery Reference](#)
- [AngularJS Reference](#)

Top 10 Examples

- [HTML Examples](#)
- [CSS Examples](#)
- [JavaScript Examples](#)
- [W3.CSS Examples](#)
- [HTML DOM Examples](#)
- [PHP Examples](#)
- [ASP Examples](#)
- [jQuery Examples](#)
- [Angular Examples](#)
- [XML Examples](#)

Web Certificates

- [HTML Certificate](#)
- [CSS Certificate](#)
- [JavaScript Certificate](#)
- [jQuery Certificate](#)
- [PHP Certificate](#)
- [Bootstrap Certificate](#)
- [XML Certificate](#)

W3Schools is optimized for learning, testing, and training. Examples might be simplified to improve reading and basic understanding. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using this site, you agree to have read and accepted our terms of use, cookie and privacy policy. Copyright 1999-2017 by Refsnes Data. All Rights Reserved.

Powered by W3.CSS.

