**CSS3 Flexbox**

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

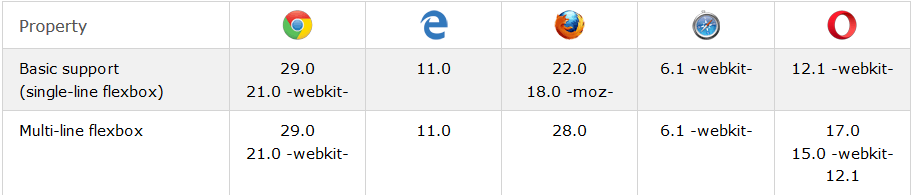
Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

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.



## CSS3 Flexbox Concepts

Flexbox consists of flex containers and flex items.

A flex container is declared by setting the display property of an element to either flex (rendered as a block) or inline-flex (rendered as inline).

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>

It is also possible to change the direction of the flex line.

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;  
}

## 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:

### Example

.flex-container {  
    display: -webkit-flex;  
    display: flex;  
    -webkit-flex-direction: row-reverse;  
    flex-direction: row-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;  
}

## 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
* flex-end - Items are positioned at the bottom of the container
* center - Items are positioned at the center of the container (vertically)
* 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;  
}

## 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.

The possible values are as follows:

* 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;  
}

## 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;  
}

## Flex Item Properties

### Ordering

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;  
}

### 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;  
}

### Perfect Centering

In the following example we will solve an almost daily problem: perfect centering.

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;  
}

### 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 {  
    -webkit-align-self: baseline;  
    align-self: baseline;  
}  
  
.item5 {  
    -webkit-align-self: stretch;  
    align-self: stretch;  
}

### 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;  
}