CSS Flexbox (Flexible Box Layout)

CSS Flexbox is short for the CSS Flexible Box Layout module.

Flexbox is a layout model for arranging items (horizontally or vertically) within a container, in a flexible and responsive way.

Flexbox makes it easy to design a flexible and responsive layout, without using float or positioning.

CSS Flexbox Components

A flexbox always consists of:

- **A Flex Container** - The parent (container) element, where the display property is set to flex or inline-flex
- **One or more Flex Items** - The direct children of the flex container automatically becomes flex items

---

A Flex Container with Three Flex Items

Example

A flex container with three flex items:

```
<html>
<head>
<style>
.container {
  display: flex;
  background-color: DodgerBlue;
}

.container div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
</style>
</head>
<body>

<div class="container">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

</body>
</html>
```

CSS Flex Container (What & Why)

A **flex container** is a parent element that uses **Flexbox layout** to arrange its child items **easily in rows or columns**.

Use:

- To create layouts (navbar, cards, center content, responsive design)
- Easier than float and positioning

```
.container {
  display: flex;
}
```

1️⃣ display: flex / inline-flex

What it does:

Makes the element a flex container.

```
.container {
  display: flex;
}
```

Use: Enables flexbox layout for children.

2️⃣ flex-direction

What it does:

Decides **direction of items**.

Values & Example:

```
.container {
  display: flex;
  flex-direction: row; /* default */
}
```

Value	**Meaning**
row	Left → Right
column	Top → Bottom
row-reverse	Right → Left
column-reverse	Bottom → Top

Use: Control layout direction (vertical menu, horizontal cards).

3️⃣ flex-wrap

What it does:

Controls whether items go to **next line** or not.

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

Value	Meaning
nowrap	One line only (default)
wrap	Move to next line
wrap-reverse	Wrap in reverse

Use: Responsive layouts (cards wrap on small screens).

4️⃣ flex-flow (Shortcut)

What it does:

Shortcut for flex-direction + flex-wrap.

```
.container {
  display: flex;
  flex-flow: row wrap;
}
```

Use: Cleaner and shorter code.

5️⃣ justify-content (Horizontal alignment)

What it does:

Aligns items **along main axis (X-axis)**.

```
.container {
  display: flex;
  justify-content: center;
}
```

Value	**Use**
flex-start	Start (default)
flex-end	End
center	Center
space-between	Space between items
space-around	Space around
space-evenly	Equal space everywhere

Use: Navbar spacing, centering buttons.

6️⃣ align-items (Vertical alignment)

What it does:

Aligns items **along cross axis (Y-axis)**.

```
.container {
  display: flex;
  height: 200px;
  align-items: center;
}
```

Value	**Meaning**
flex-start	Top
flex-end	Bottom
center	Middle
stretch	Fill height (default)

Value	**Meaning**
baseline	Text baseline

Use: Vertically center content.

7️⃣ align-content (Multiple lines)

What it does:

Aligns **rows of items**, not single items.
Works only when flex-wrap: wrap.

```
.container {
 display: flex;
 flex-wrap: wrap;
 align-content: space-between;
}
```

Value	**Use**
center	Center lines
flex-start	Top
flex-end	Bottom
space-between	Equal gaps
space-around	Space around lines
space-evenly	Equal spacing

Use: Control spacing between rows in grid-like layouts.

True Centering (Most Important)

Perfect center (horizontal + vertical):

```
.container {
  display: flex;
  height: 300px;
  justify-content: center;
  align-items: center;
}
```

Use: Login box, modal, loader, card center.

Summary Table (Easy to Remember)

Property	**Use**
display	Enable flex
flex-direction	Row or column
flex-wrap	Wrap items
flex-flow	Direction + wrap
justify-content	Horizontal alignment
align-items	Vertical alignment
align-content	Row alignment

CSS Flex Items (What are they?)

All **direct children** of a flex container automatically become **flex items**.

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
.flex-container {
  display: flex;
}
```

Here, 1, 2, 3 are **flex items**.

1️⃣ order property

What it does:

Changes the **display order** of flex items (without changing HTML).

- Default value: 0
- Smaller number → appears first

Example:

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```

Result order on screen: 4 2 1 3

Use:
Navbar items rearrange, responsive design without touching HTML.

2️⃣ flex-grow

What it does:

Controls **how much an item grows** compared to others.

- Default: 0
- Higher number = grows more

Example:

```
<div class="flex-container">

  <div style="flex-grow: 1">1</div>

  <div style="flex-grow: 1">2</div>

  <div style="flex-grow: 4">3</div>

</div>
```

Item **3** becomes much wider.

Use:
Main content bigger than sidebar.

3️⃣ flex-shrink

What it does:

Controls **how much an item shrinks** when space is less.

- Default: 1
- Higher value = shrinks more

Example:

```
<div class="flex-container">

  <div>1</div>

  <div>2</div>

  <div style="flex-shrink: 2">3</div>

  <div>4</div>

</div>
```

Item **3** shrinks faster.

Use:
Prevent important items from shrinking.

4️⃣ flex-basis

What it does:

Sets the **initial size** of a flex item.

Example:

```
<div class="flex-container">

  <div>1</div>

  <div>2</div>

  <div style="flex-basis: 250px">3</div>

  <div>4</div>

</div>
```

Item **3** starts at 250px.

Use:
Set default width of cards or columns.

5️⃣ flex (Shortcut)

What it does:

Shortcut for:

flex-grow | flex-shrink | flex-basis

Example:

```
<div class="flex-container">

  <div>1</div>
  <div>2</div>

  <div style="flex: 1 0 150px">3</div>

  <div>4</div>

</div>
```

Means:

- grow = 1
- shrink = 0
- size = 150px

Use:
Cleaner, professional CSS.

6️⃣ align-self

What it does:

Aligns **one specific item**, overriding align-items.

Example 1 (center one item):

```
<div class="flex-container">

  <div>1</div>

  <div>2</div>

  <div style="align-self: center">3</div>

  <div>4</div>

</div>
```

Example 2 (top & bottom):

```
<div class="flex-container">

  <div>1</div>

  <div style="align-self: flex-start">2</div>
  <div style="align-self: flex-end">3</div>
  <div>4</div>
</div>
```

Use:
Highlight one card, special alignment.

Quick Summary Table

Property	**Use**
order	Change item position
flex-grow	Increase size
flex-shrink	Reduce size
flex-basis	Initial width
flex	Shortcut
align-self	Align single item

Responsive Flexbox

You learned from the CSS Media Queries chapter that you can use media queries to create different layouts for different screen sizes and devices.

For example, if you want to create a three-column layout for large screen sizes, and a one-column layout for small screen sizes (such as phones), you can change the flex-direction from row to column at a specific breakpoint (600px in the example below):

Resize the browser window to see the effect.

Example

```
.flex-container {
  display: flex;
  flex-direction: row;
}

.flex-item {
  background-color: #f1f1f1;
  padding: 10px;
  font-size: 30px;
  text-align: center;
  width: 100%;
}

/* Make a one column-layout instead of three-column layout */
@media (max-width: 600px) {
  .flex-container {
    flex-direction: column;
  }
}
```

Another way is to change the percentage of the flex property of the flex items to create different layouts for different screen sizes. Note that we also have to include flex-wrap: wrap; on the flex container for this example to work:

Example

```
.flex-container {
  display: flex;
  flex-wrap: wrap;
}

.flex-item {
  background-color: #f1f1f1;
  padding: 10px;
  text-align: center;
  font-size: 30px;
  flex: 33.3%;
}
```

```
/* Make a one column-layout instead of a three-column layout */
@media (max-width: 600px) {
  .flex-item {
    flex: 100%;
  }
}
```