

SWEN1005

MOBILE WEB PROGRAMMING

Session Four

- Body
- Table vs Div vs Flexbox
- Lists

The <body>

- The <body> tag defines the document's body.
- The <body> element contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

The <body> attributes

- Background, text, bgcolor etc. are not supported by HTML5.
- However, global attributes and global events are supported

Global Attributes

- HTML attributes give elements meaning and context.
- The global attributes below can be used on any HTML element.

Example – accesskey

```
<a href="https://www.w3schools.com/html5" accesskey="h">HTML5</a><br>  
<a href="https://www.w3schools.com/css3" accesskey="c">CSS3</a>
```

- Provides an alternative way to activate a link

Example – accesskey

```
<!DOCTYPE html>
<html>
<body>
<a href="https://www.w3schools.com/html" accesskey="h">HTML tutorial</a><br>
<a href="https://www.w3schools.com/css" accesskey="c">CSS tutorial</a>
<p>The accesskey attribute is a shortcut to activate/focus an element.</p>
<p><strong>Note:</strong> The shortcut is varying in different browsers:</p>
<ul>
  <li>IE, Chrome, Safari, Opera 15+: [ALT] + <em>accesskey</em></li>
  <li>Opera prior version 15: [SHIFT] [ESC] + <em>accesskey</em></li>
  <li>Firefox: [ALT] [SHIFT] + <em>accesskey</em></li>
</ul>
</body>
</html>
```

Other Global Attributes

Attribute	Description
<u>accesskey</u>	Specifies a shortcut key to activate/focus an element
<u>class</u>	Specifies one or more class names for an element
<u>contenteditable</u>	Specifies whether the content of an element is editable or not
<u>contextmenu</u>	Specifies a context menu for an element. The context menu appears when a user right-clicks on the element
<u>data-*</u>	Used to store custom data private to the page or application
<u>dir</u>	Specifies the text direction for the content in an element
<u>draggable</u>	Specifies whether an element is draggable or not

Other Global Attributes

Attribute	Description
<u>dropzone</u>	Specifies whether the dragged data is copied, moved, or linked, when dropped
<u>hidden</u>	Specifies that an element is not yet, or is no longer, relevant
<u>id</u>	Specifies a unique id for an element
<u>lang</u>	Specifies the language of the element's content
<u>spellcheck</u>	Specifies whether the element is to have its spelling and grammar checked
<u>style</u>	Specifies an inline CSS style for an element
<u>tabindex</u>	Specifies the tabbing order of an element
<u>title</u>	Specifies extra information about an element
<u>translate</u>	Specifies whether the content of an element should be translated or not

Global Events

- Since HTML 4 an event is the ability to trigger actions in a browser, like starting a JavaScript when a user clicks on an element.
- These elements include:
 - Window
 - Mouse
 - Keyboard
 - Drag
 - Clipboard
 - Media
 - Miscellaneous

Global Events

Attribute	Value	Description
<u>onafterprint</u>	<i>script</i>	Script to be run after the document is printed
<u>onbeforeprint</u>	<i>script</i>	Script to be run before the document is printed
<u>onbeforeunload</u>	<i>script</i>	Script to be run when the document is about to be unloaded
<u>onerror</u>	<i>script</i>	Script to be run when an error occurs
<u>onhashchange</u>	<i>script</i>	Script to be run when there has been changes to the anchor part of the a URL
<u>onload</u>	<i>script</i>	Fires after the page is finished loading

Global Events

Attribute	Value	Description
onmessage	<i>script</i>	Script to be run when the message is triggered
<u>onoffline</u>	<i>script</i>	Script to be run when the browser starts to work offline
<u>ononline</u>	<i>script</i>	Script to be run when the browser starts to work online
onpagehide	<i>script</i>	Script to be run when a user navigates away from a page
<u>onpageshow</u>	<i>script</i>	Script to be run when a user navigates to a page
onpopstate	<i>script</i>	Script to be run when the window's history changes
<u>onresize</u>	<i>script</i>	Fires when the browser window is resized
onstorage	<i>script</i>	Script to be run when a Web Storage area is updated
<u>onunload</u>	<i>script</i>	Fires once a page has unloaded (or the browser window closed)

Global Event Examples

- ``
- ``
`<script>`
 `function loadImage() {`
 `alert("Image is loaded");`
 `}`
`</script>`

Session Five

TABLES VS DIVISION

What is a table?

- An HTML table is defined with the `<table>` tag.
- Each table row is defined with the `<tr>` tag.
- A table header is defined with the `<th>` tag. By default, table headings are bold and centered.
- A table data/cell is defined with the `<td>` tag. The `<td>` elements are the data containers of the table. They can contain all sorts of HTML elements; text, images, lists, other tables, etc.

Table

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Table - HTML

```
<table style="width:100%">
```

```
<tr>
```

```
<th>Firstname</th>
```

```
<th>Lastname</th>
```

```
<th>Age</th>
```

```
</tr>
```

```
<tr>
```

```
<td>Jill</td>
```

```
<td>Smith</td>
```

```
<td>50</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Eve</td>
```

```
<td>Jackson</td>
```

```
<td>94</td>
```

```
</tr>
```

```
</table>
```

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Table – colspan

Name	Telephone	
Bill Gates	55577854	55577855

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
th, td {
    padding: 5px;
    text-align: left;
}
</style>
</head>
<body>

<h2>Cell that spans two columns:</h2>
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>Bill Gates</td>
    <td>55577854</td>
    <td>55577855</td>
  </tr>
</table>

</body>
</html>
```

Table - rowspan

Name:	Bill Gates
Telephone:	55577854
	55577855

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
th, td {
    padding: 5px;
    text-align: left;
}
</style>
</head>
<body>
<h2>Cell that spans two rows:</h2>
<table style="width:100%">
    <tr>
        <th>Name:</th>
        <td>Bill Gates</td>
    </tr>
    <tr>
        <th rowspan="2">Telephone:</th>
        <td>55577854</td>
    </tr>
    <tr>
        <td>55577855</td>
    </tr>
</table>

</body>
</html>
```

<div> - division

- The <div> tag defines a division or a section in an HTML document.
- The <div> tag is used to group block-elements to format them with CSS.
- The <div> element is very often used together with CSS, to layout a web page.
- By default, browsers always place a line break before and after the <div> element. However, this can be changed with CSS.

<div> - inline CSS

- <!DOCTYPE html>
- <html>
- <body>
- <p>This is some text.</p>
- <div style="color:#0000FF">
- <h3>This is a heading in a div element</h3>
- <p>This is some text in a div element.</p>
- </div>
- <p>This is some text.</p>
- </body>
- </html>

This is some text.

This is a heading in a div element

This is some text in a div element.

This is some text.


```

<!DOCTYPE html>
<html>
<body>

<table cellpadding="0" cellspacing="0" border="1">
  <tr>
    <td colspan="3" height="120px">Header</td>
  </tr>
  <tr>
    <td class="menu" valign="top">Menu</td>
    <td class="content" valign="top">Content</td>
    <td class="aSide" valign="top">Aside</td>
  </tr>
  <tr>
    <td colspan="3">Footer</td>
  </tr>
</table>

</body>
</html>

```

Header		
Menu	Content	Aside
Footer		

<div> Layout

```
<!DOCTYPE html>
<html>
<body>

<div id="header">Header</div>
<div id="menu">Menu</div>
<div id="content">Content</div>
<div id="aSide">Aside</div>
<div id="footer">Footer</div>

</body>
</html>
```

Header
Menu
Content
Aside
Footer

Which would you choose and why?

Session Five

FLEXBOX

Flexbox (or flexible box layout)

- Is a new box model optimized for UI layout.
- It is one of the first CSS modules designed for actual layout
- it makes a lot of tasks much easier, or even possible at all.
- Its repertoire includes:
 - The simple centering of elements (both horizontally and vertically)
 - The expansion and contraction of elements to fill available space
 - Source code independent layout
 - Among others

Flexbox

- Its 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.
- The flexible box model provides an improvement over the block model
 - It does not use floats, nor do the flex container's margins collapse with the margins of its contents

Flexbox Concepts

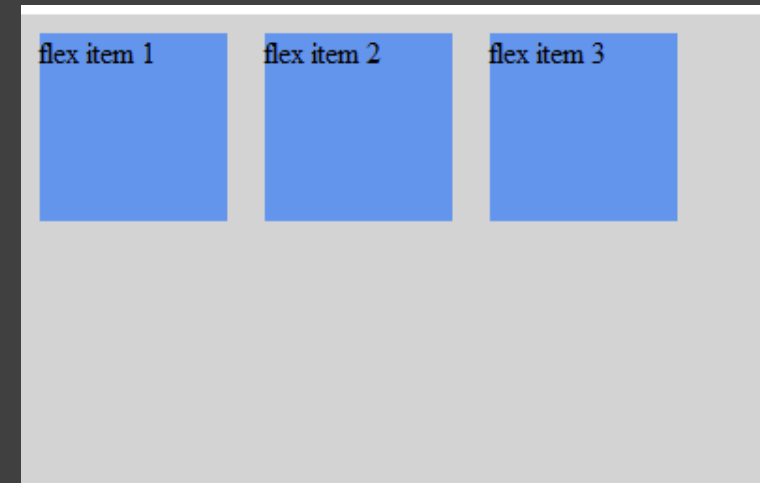
- Flexbox consists of flex containers and flex items.
- A flex container is declared by setting the display property of an element to either flex (rendered as a block) or inline-flex (rendered as inline).
- Inside a flex container there is one or more flex items.

Flexbox - Example

```
<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>
```

```
<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>
```



These items are positioned by default: along the horizontal flex line, left to right

Flexbox Concepts

- 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.

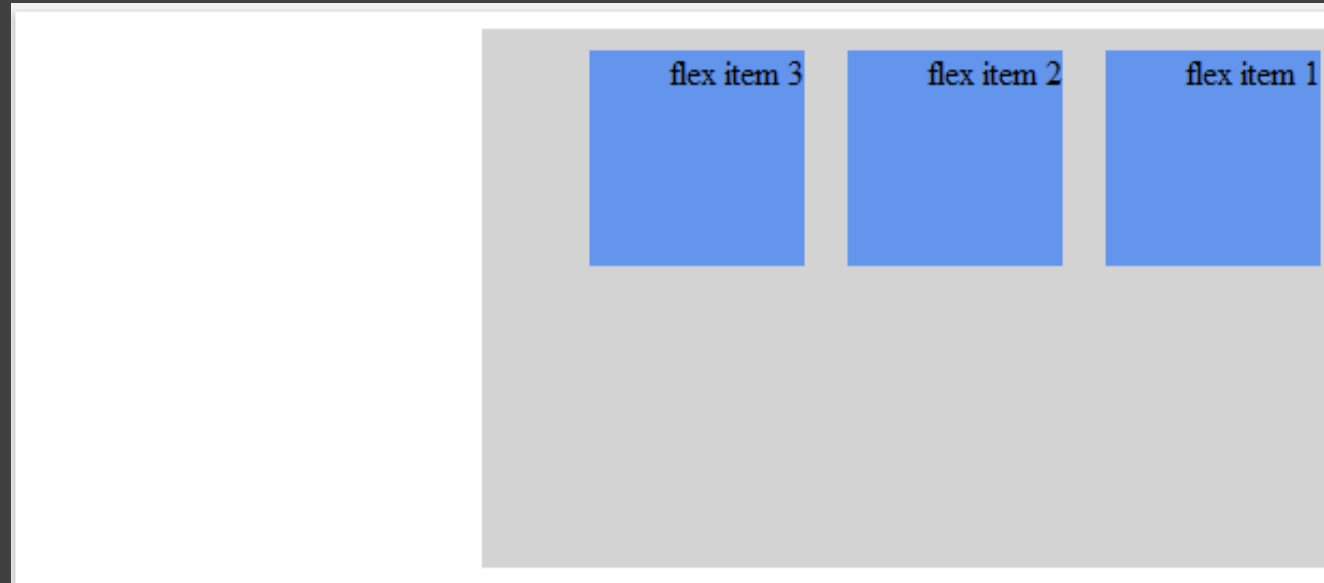
Flexbox - rtl

```
<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;  
}
```

Flexbox - rtl



```
<style>  
body {  
  direction: rtl;  
}  
  
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  width: 400px;  
  height: 200px;  
}
```

```
<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>
```

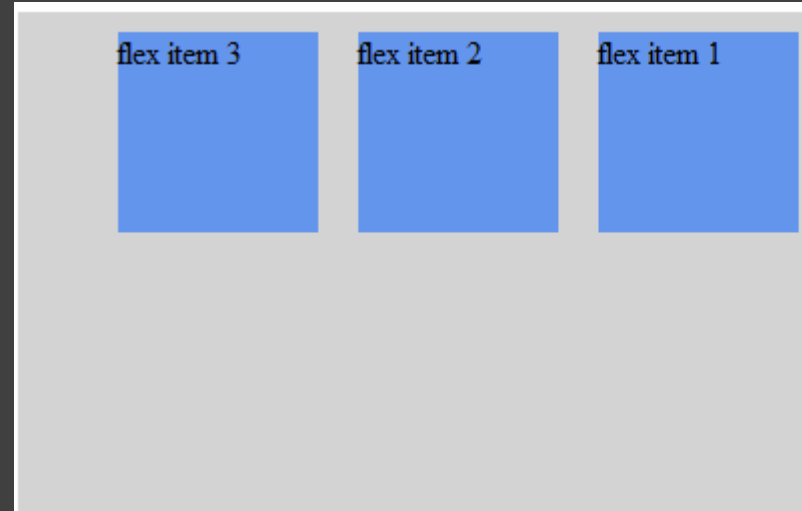

Flexbox Direction

Flex Direction

- Specifies the direction of the flexible items inside the flex container
- 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


Flex direction: row-reverse

```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-direction: row-reverse;
  flex-direction: row-reverse;
  width: 400px;
  height: 250px;
```



Flex direction: column

```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-direction: column;
  flex-direction: column;
  width: 400px;
  height: 250px;
}
```



A diagram illustrating a flex container with a light gray background. Three blue rectangular items are stacked vertically on the left side of the container. Each item is labeled with its respective text: 'flex item 1', 'flex item 2', and 'flex item 3'.

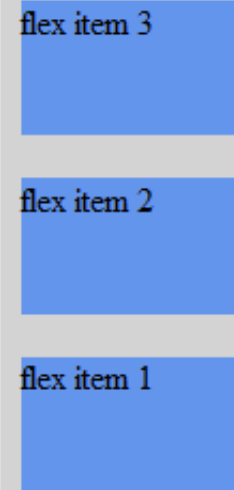
flex item 1

flex item 2

flex item 3

Flex direction: column reverse

```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-direction: column-reverse;
  flex-direction: column-reverse;
  width: 400px;
  height: 250px;
}
```



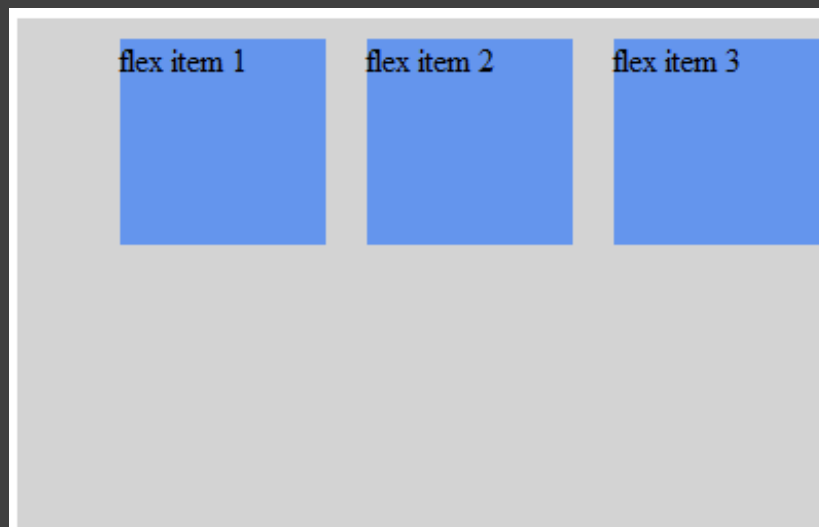
A diagram illustrating a flex container with a light gray background. Three blue rectangular flex items are stacked vertically on the left side of the container. The items are labeled from top to bottom: "flex item 3", "flex item 2", and "flex item 1". This demonstrates the "column-reverse" flex direction, where items are ordered from the end of the list to the beginning.

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:
 - 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

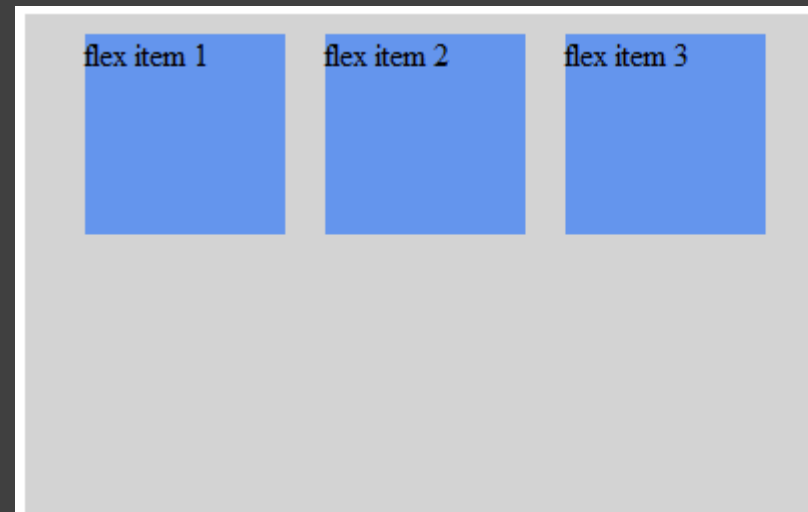
justify-content – flex-end

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



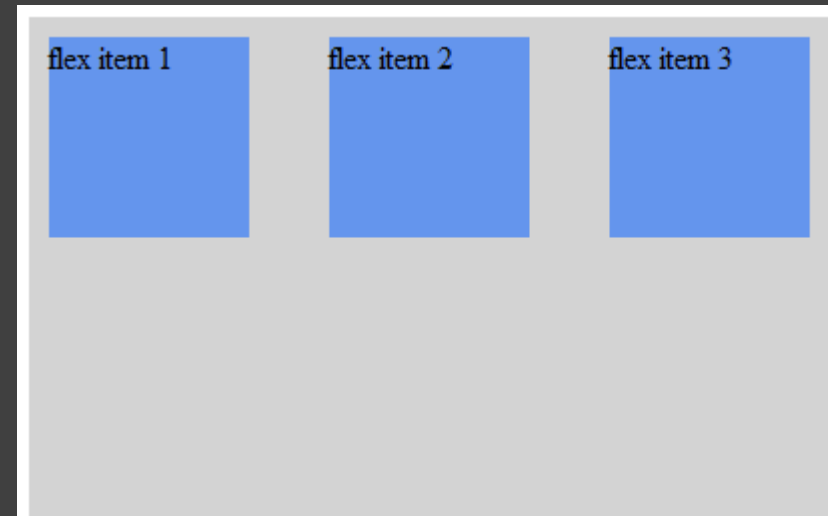
justify-content – center

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



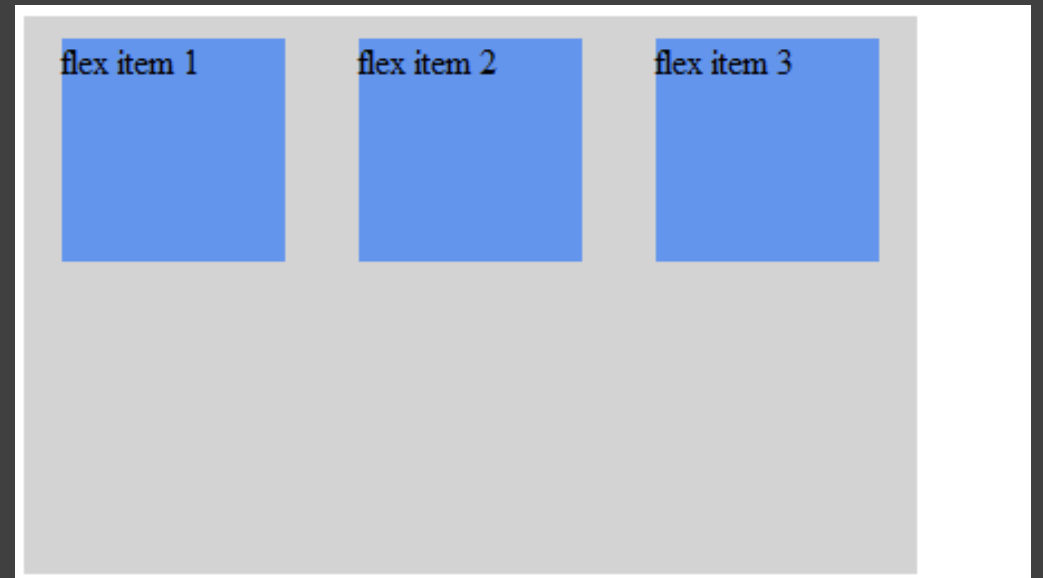
justify-content – space-between

```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-justify-content: space-between;
  justify-content: space-between;
  width: 400px;
  height: 250px;
}
```



justify-content – space-around

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

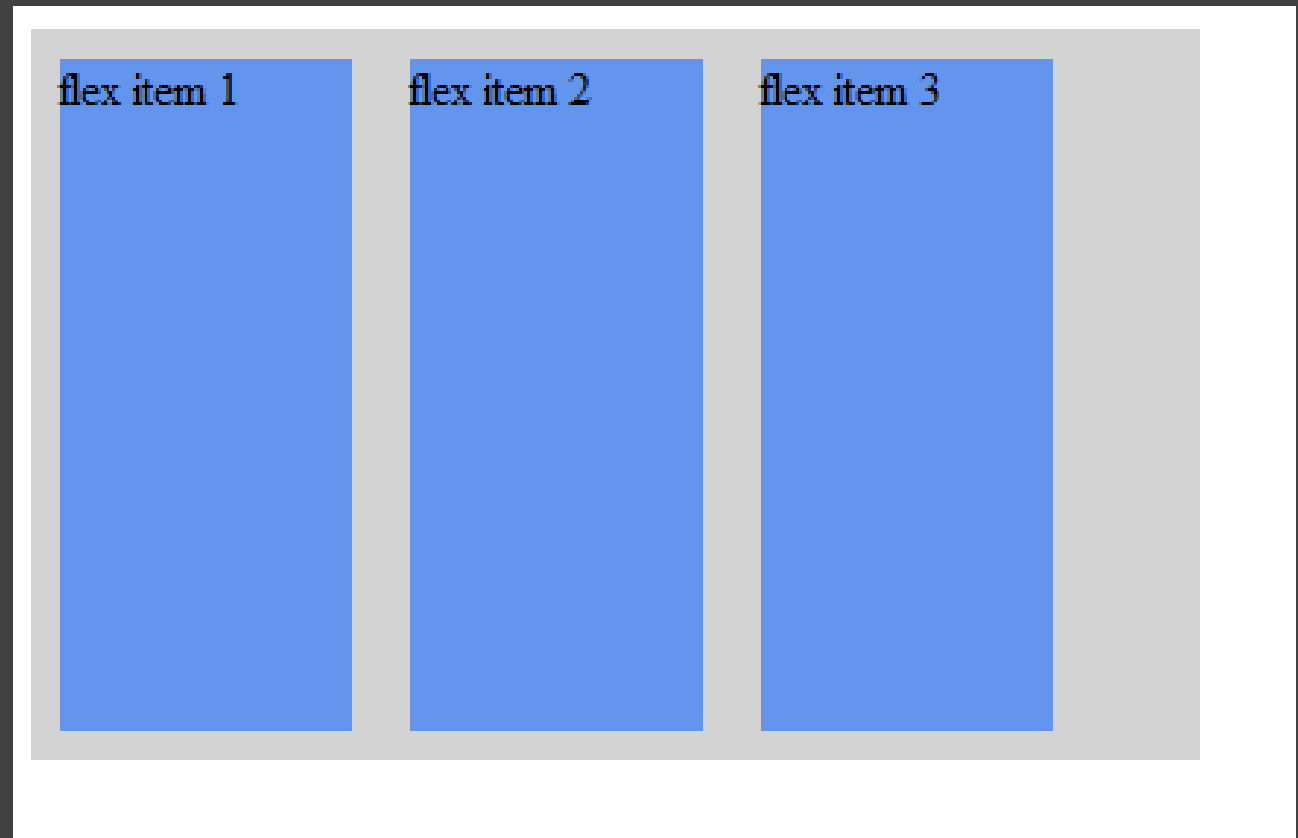


The align-items Property

- 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

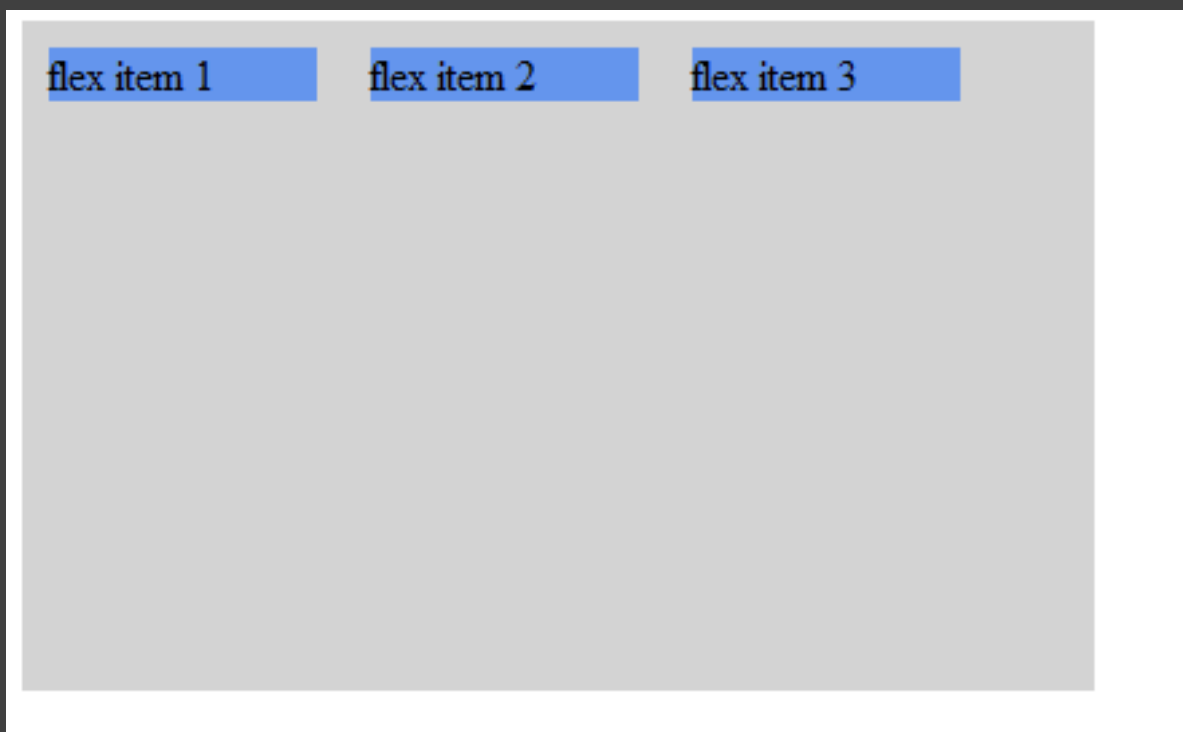
align-items: stretch

```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-align-items: stretch;
  align-items: stretch;
  width: 400px;
  height: 250px;
}
```



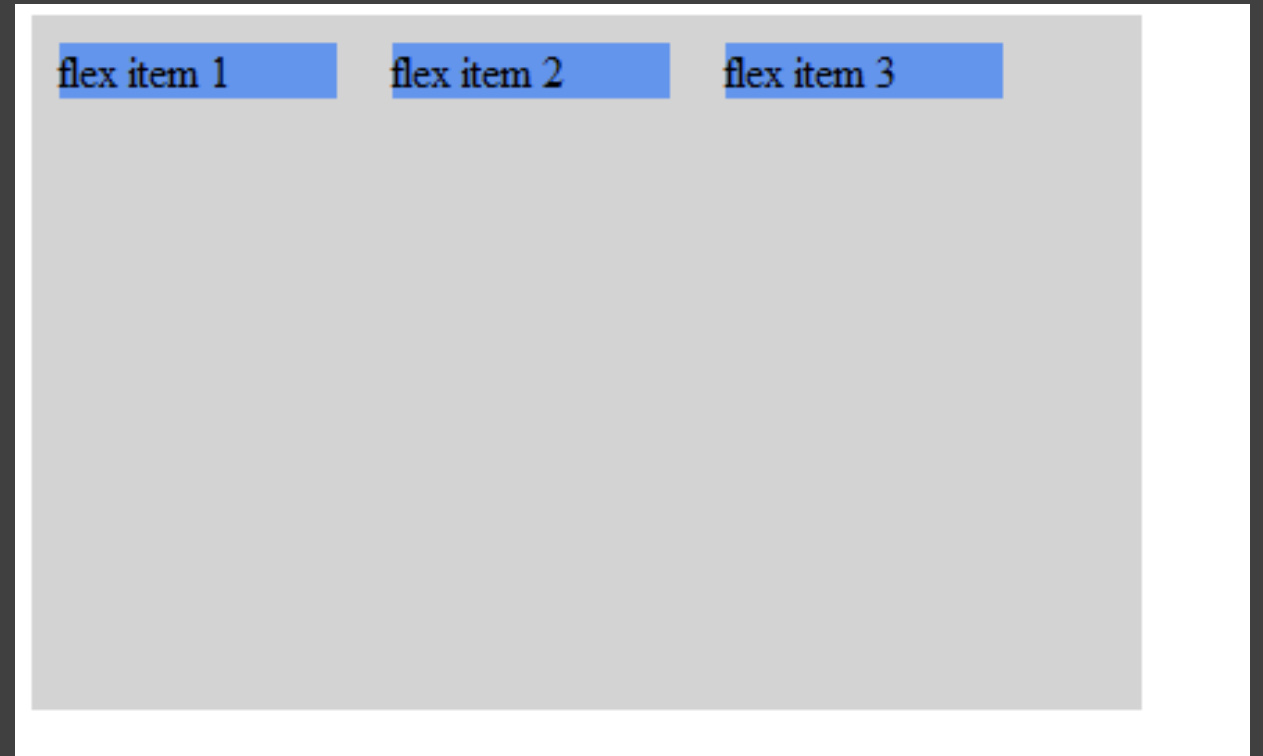
align-items: flex-start

```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-align-items: flex-start;
  align-items: flex-start;
  width: 400px;
  height: 250px;|
```



align-items: flex-start

```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-align-items: flex-start;
  align-items: flex-start;
  width: 400px;
  height: 250px;|
}
```

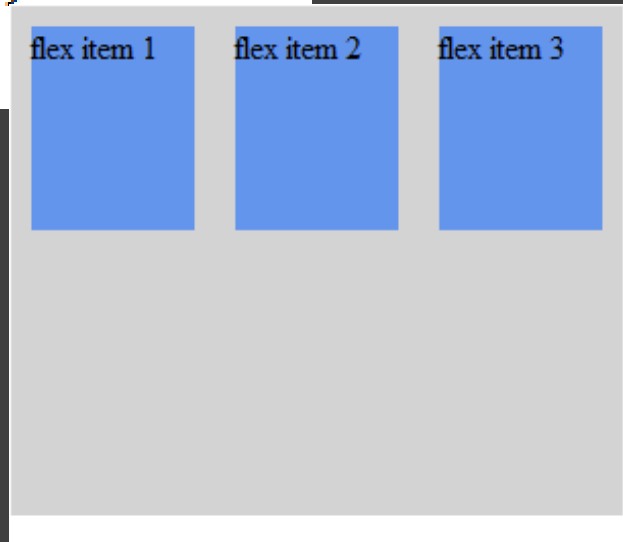


The flex-wrap Property

- 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 flex-wrap Property

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

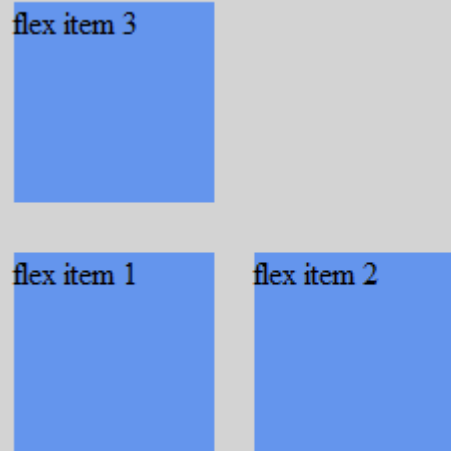


```
<style>  
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-wrap: wrap;  
  flex-wrap: wrap;  
  width: 300px;  
}
```



The `flex-wrap` Property

```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-wrap: wrap-reverse;
  flex-wrap: wrap-reverse;
  width: 300px;
  height: 250px;
```



```
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-wrap: wrap;
  flex-wrap: wrap;
  width: 300px;
```



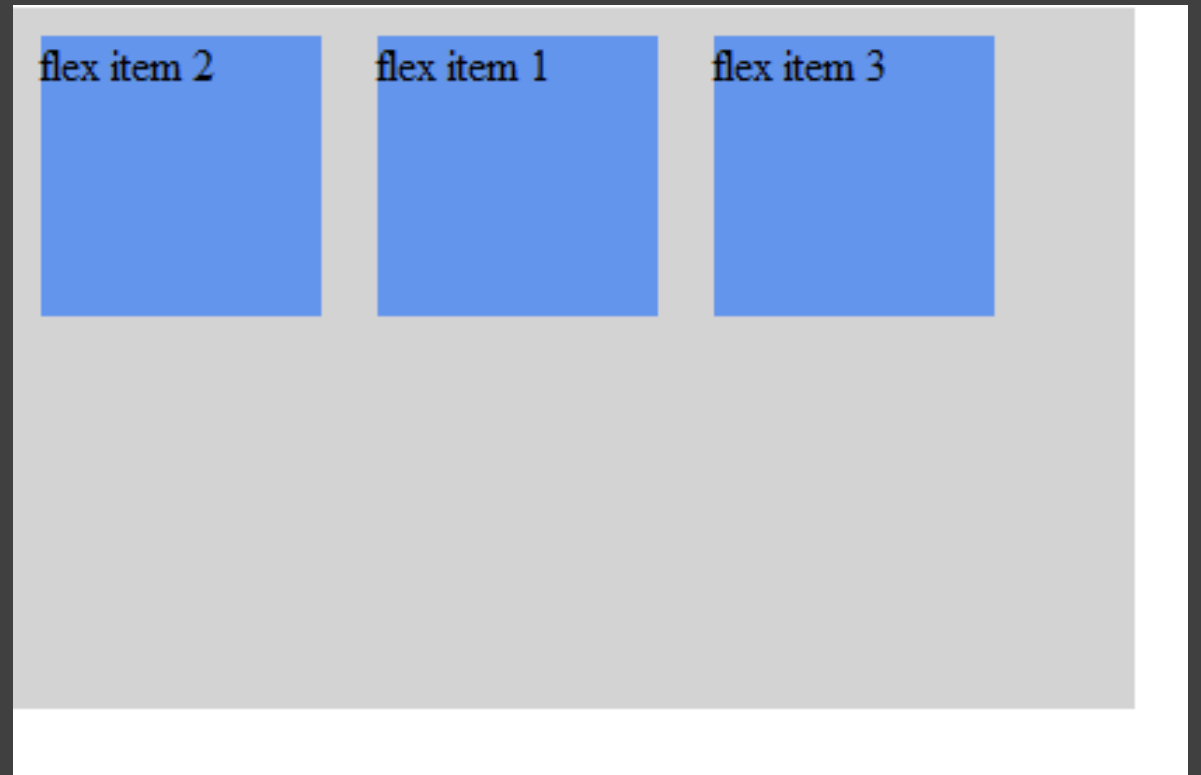
Flex Item Properties - Ordering

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

```
<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;
}

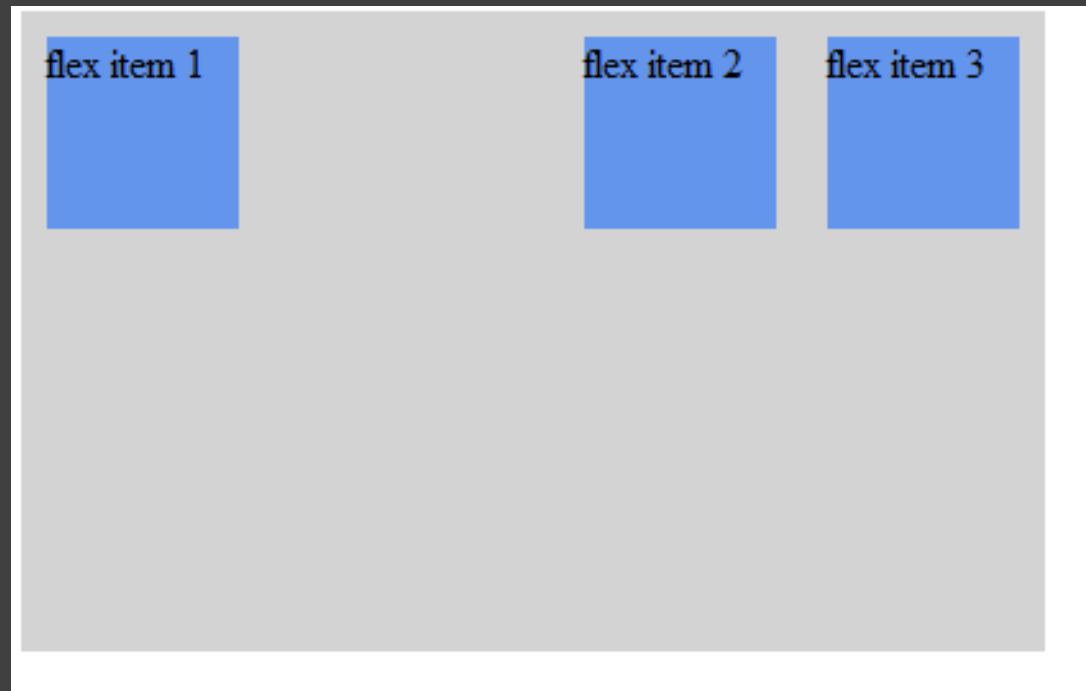
.first {
  -webkit-order: -1;
  order: -1;
}
</style>
```



Margin

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

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

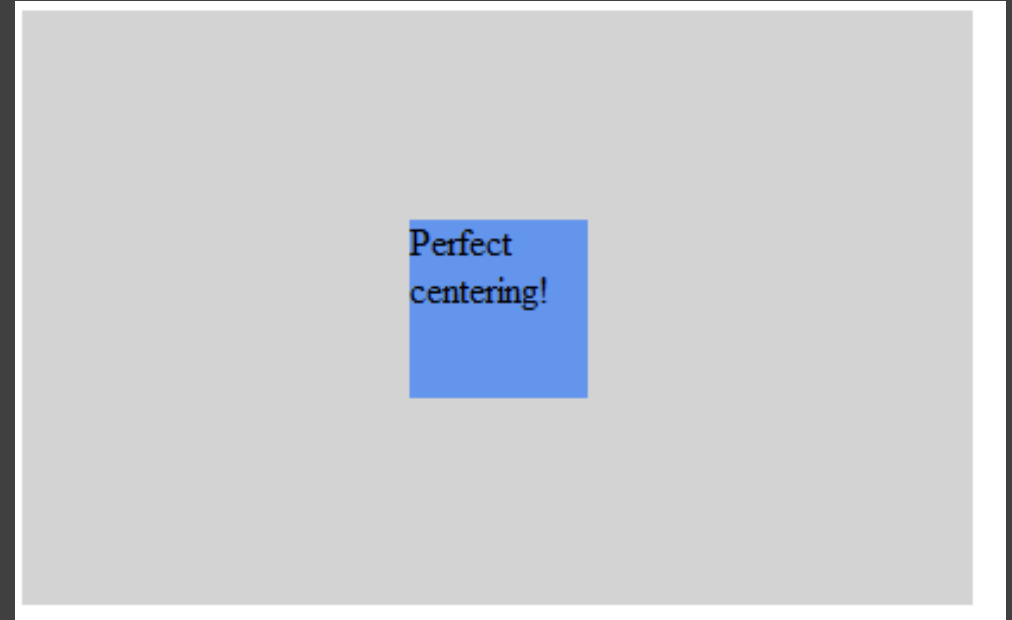


Perfect Centering

- 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:

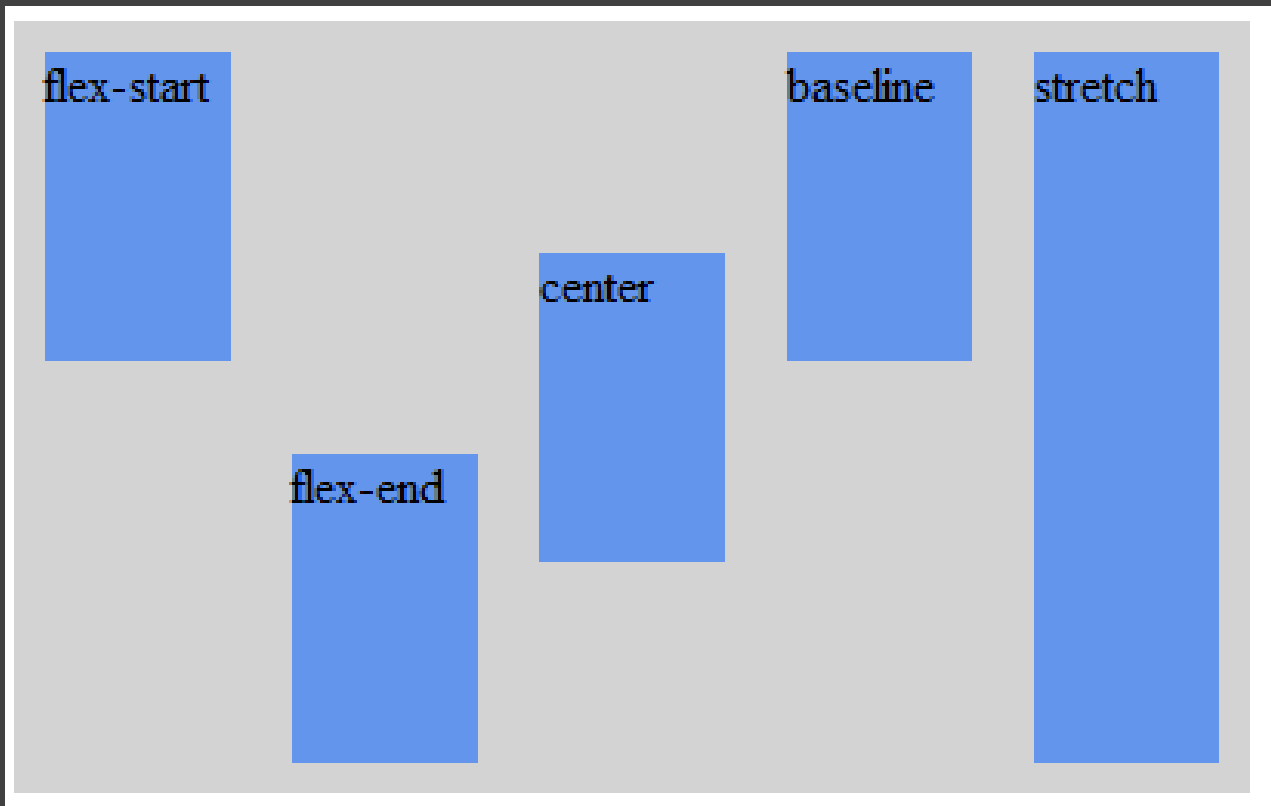
Perfect Centering

```
.flex-item {  
  background-color: cornflowerblue;  
  width: 75px;  
  height: 75px;  
  margin: auto;  
}  
</style>  
</head>  
<body>  
  
<div class="flex-container">  
  <div class="flex-item">Perfect centering!</div>  
</div>
```



align-self

- 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



```
.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;
}
</style>
</head>
<body>

<div class="flex-container">
  <div class="flex-item item1">flex-start</div>
  <div class="flex-item item2">flex-end</div>
  <div class="flex-item item3">center</div>
  <div class="flex-item item4">baseline</div>
  <div class="flex-item item5">stretch</div>
</div>
```


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 $\frac{2}{4}$ of the free space, and the other two flex items will consume $\frac{1}{4}$ of the free space each:

```
}

.flex-item {
  background-color: cornflowerblue;
  margin: 10px;
}

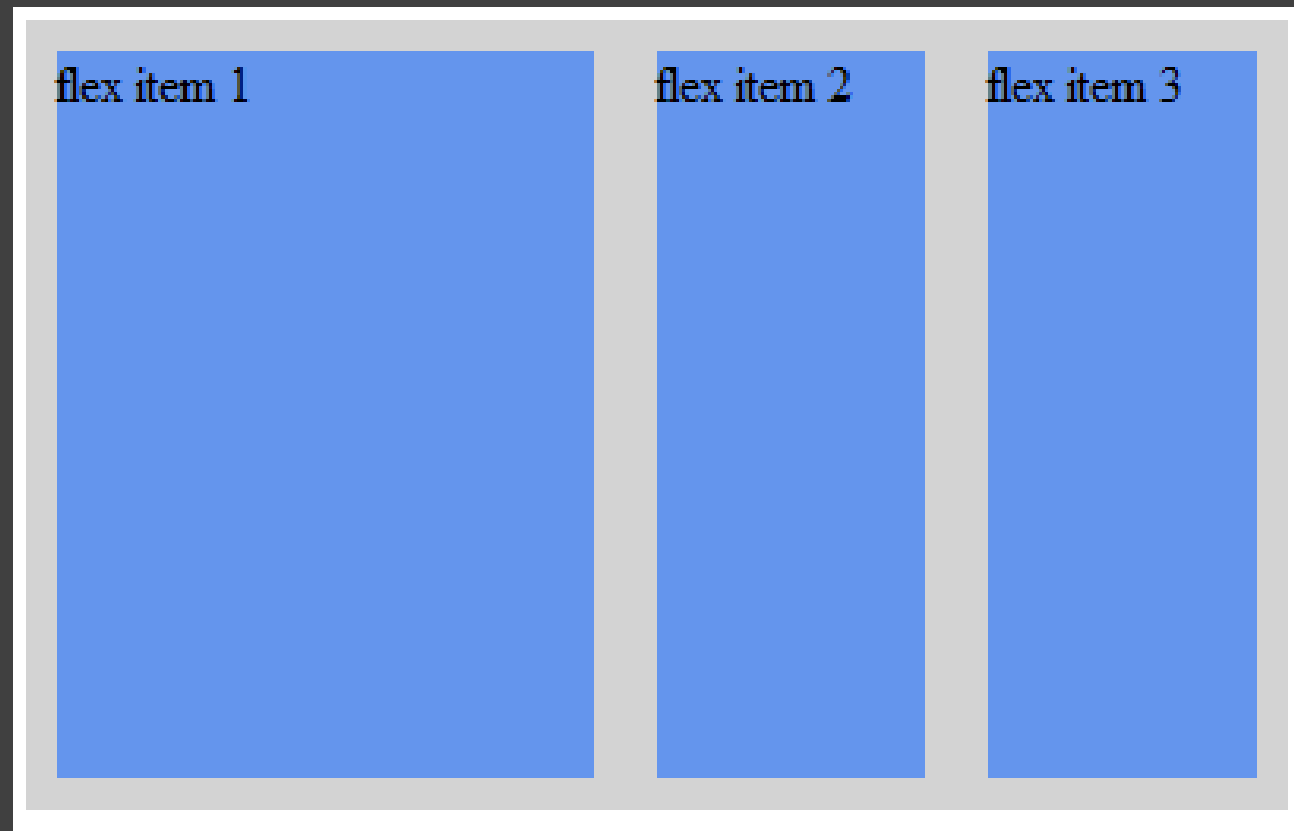
.item1 {
  -webkit-flex: 2;
  flex: 2;
}

.item2 {
  -webkit-flex: 1;
  flex: 1;
}

.item3 {
  -webkit-flex: 1;
  flex: 1;
}

</style>
</head>
<body>

<div class="flex-container">
  <div class="flex-item item1">flex item 1</div>
  <div class="flex-item item2">flex item 2</div>
  <div class="flex-item item3">flex item 3</div>
</div>
```



Quick Review

- Tables
- Division block element
- Flex boxes

Questions?
