

## Day 25

# "Web Development + Security"

### More on CSS Selectors:

CSS Cheat Sheet:

Selector Type	Syntax	Example	Meaning / What It Selects
Universal Selector	*	* { margin: 0; }	Selects <b>all elements</b>
Element Selector	tag	p { color: blue; }	Selects <b>all &lt;p&gt;</b> elements
Class Selector	.classname	.intro { color: green; }	Selects all elements with class="intro"
ID Selector	#idname	#main { background: yellow; }	Selects element with id="main"
Group Selector	A, B	h1, h2 { color: red; }	Applies same style to <b>multiple elements</b>
Descendant Selector	A B	div p { color: blue; }	Selects <p> <b>inside</b> <div>
Child Selector	A > B	div > p { color: orange; }	Selects <b>direct child</b> <p> of <div>
Adjacent Sibling	A + B	h1 + p { color: purple; }	Selects <p> <b>immediately after</b> <h1>
General Sibling	A ~ B	h1 ~ p { color: gray; }	Selects <b>all</b> <p> <b>after</b> <h1> (siblings)
Attribute Selector	[attr]	[title] { color: pink; }	Elements <b>having</b> title attribute
	[attr=value]	[type="text"]	Elements with attribute <b>equal to</b> given value
	[attr^=value]	[class^="btn"]	Attribute <b>starts with</b> value
	[attr\$=value]	[src\$=".jpg"]	Attribute <b>ends with</b> value
	[attr*=value]	[class*="nav"]	Attribute <b>contains</b> value

Selector Type	Syntax	Example	Meaning / What It Selects
Pseudo-class	:pseudo-class	a:hover { color: red; }	Defines a <b>state</b> of an element
Common Pseudo-classes	:hover, :focus, :active, :nth-child(), :first-child, :last-child	—	Used for interactivity and element positions
Pseudo-element	::before, ::after	p::before { content: "👉"; }	Adds <b>content before or after</b> an element
Negation Selector	:not(selector)	p:not(.special)	Selects <p> <b>except</b> those with .special

## CSS Flexbox:

### What is Flex box?

Flexbox (Flexible Box) is a CSS layout module that allows you to align and distribute space among items in a container, even when their size is unknown or dynamic. Think of it as a magic tool to control layouts easily, without floats or complicated positioning.

To understand this, we will first see how without it things work?

A basic example of <div>s: clearly, we can see that all the <div> get one above other, like in a column.



```
flex.html > html > body > main > div.container > div.item
1 <html lang="en">
2   <head>
3     <style>
4       .container{
5         border: 2px solid red;
6       }
7       .item{
8         height: 12px;
9         width: 12px;
10        background-color: blue;
11        margin: 4px;
12        border: 2px solid black;
13     }
14   </style>
15 </head>
16 <body>
17   <main>
18     <div class="container">
19       <div class="item"></div>
20       <div class="item"></div>
21       <div class="item"></div>
22       <div class="item"></div>
23       <div class="item"></div>
24     </div>
25   </main>
26 </body>
27 </html>
```

What if we add `display:flex` in the `.container`? We will see that all the <div> come in one row, as shown below:



```
flex.html > html > body > main
1 <html lang="en">
2   <head>
3     <style>
4       .container{
5         border: 2px solid red;
6         display: flex;
7       }
8       .item{
9         height: 12px;
10        width: 12px;
11        background-color: blue;
12        margin: 4px;
13        border: 2px solid black;
14     }
15   </style>
16 </head>
17 <body>
18   <main>
19     <div class="container">
20       <div class="item"></div>
21       <div class="item"></div>
22       <div class="item"></div>
23       <div class="item"></div>
24       <div class="item"></div>
25     </div>
26   </main>
27 </body>
28 </html>
```

## Main Flex Properties

### A. Container Properties

1. **flex-direction** → Sets the main axis (row or column)

```
.container {
  display: flex;
  flex-direction: row; /* row (default), column, row-reverse, column-reverse */
}
```

2. **justify-content** → Align items along the **main axis**

```
.container {  
  justify-content: flex-start; /* flex-start, flex-end, center, space-between, space-around, space-evenly */  
}  
}
```

3. **align-items** → Align items along the **cross axis** (perpendicular to main axis)

```
.container {  
  align-items: stretch; /* flex-start, flex-end, center, baseline, stretch */  
}  
}
```

4. **flex-wrap** → Allow items to wrap to next line

```
.container {  
  flex-wrap: wrap; /* nowrap (default), wrap, wrap-reverse */  
}  
}
```

5. **gap** → Space between items

```
.container {  
  gap: 10px; /* sets spacing between flex items */  
}  
}
```

## B. Item Properties

1. **flex-grow** → Make an item grow to fill available space

```
.item {  
  flex-grow: 1; /* default 0 */  
}  
}
```

2. **flex-shrink** → Make an item shrink if necessary

```
.item {  
  flex-shrink: 1; /* default 1 */  
}  
}
```

3. **flex-basis** → Initial size of item before growing/shrinking

```
.item {  
  flex-basis: 200px;  
}  
}
```

4. **align-self** → Override align-items for a single item

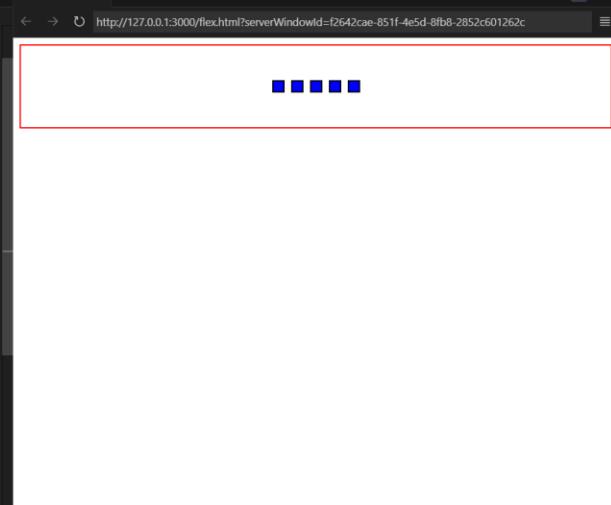
```

.item {
  align-self: center; /* auto, flex-start, flex-end, center, baseline, stretch */
}

```

Now, what to do in order to make the elements in centre?

We will use the property, justify-content: center; and align-items: center; as shown below:



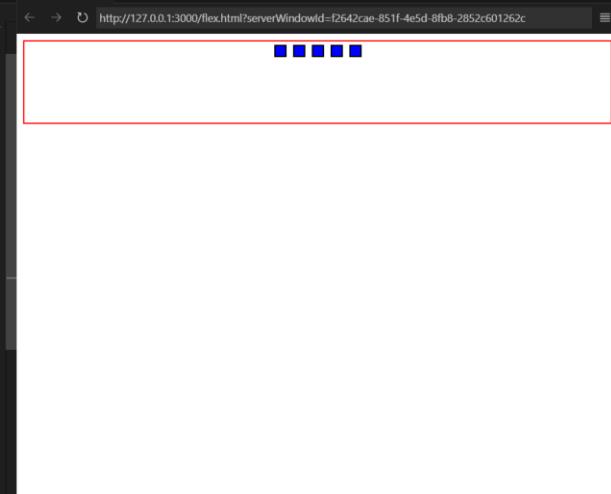
The screenshot shows a browser window displaying a page at <http://127.0.0.1:3000/flex.html>. The page contains a CSS file and a corresponding HTML output. The CSS defines a container with a height of 100px and a width of 120px. Inside the container, there are five items, each with a height of 12px and a width of 12px, colored blue. The items are centered both horizontally and vertically within the container. The browser's developer tools are visible on the left side of the screen.

```

flex.html > html > body > main > div.container > div.item
1 <html lang="en">
2   <head>
3     <style>
4       .container{
5         border: 2px solid red;
6         display: flex;
7         height: 100px;
8         justify-content: center;
9         align-items: center;
10      }
11      .item{
12        height: 12px;
13        width: 12px;
14        background-color: blue;
15        margin: 4px;
16        border: 2px solid black;
17      }
18    </style>
19  </head>
20  <body>
21    <main>
22      <div class="container">
23        <div class="item"></div>
24        <div class="item"></div>
25        <div class="item"></div>
26        <div class="item"></div>
27        <div class="item"></div>
28      </div>
29    </main>
30  </body>
31 </html>

```

In case align-item center is not there: then horizontally only it will be in centre not vertically.



The screenshot shows a browser window displaying a page at <http://127.0.0.1:3000/flex.html?serverWindowId=f2642cae-851f-4e5d-8fb8-2852c601262c>. The page contains a CSS file and a corresponding HTML output. The CSS defines a container with a height of 100px and a width of 120px. Inside the container, there are five items, each with a height of 12px and a width of 12px, colored blue. The items are centered horizontally but are not centered vertically within the container. The browser's developer tools are visible on the left side of the screen.

```

flex.html > html > body > main
1 <html lang="en">
2   <head>
3     <style>
4       .container{
5         border: 2px solid red;
6         display: flex;
7         height: 100px;
8         justify-content: center;
9         /* align-items: center; */
10      }
11      .item{
12        height: 12px;
13        width: 12px;
14        background-color: blue;
15        margin: 4px;
16        border: 2px solid black;
17      }
18    </style>
19  </head>
20  <body>
21    <main>
22      <div class="container">
23        <div class="item"></div>
24        <div class="item"></div>
25        <div class="item"></div>
26        <div class="item"></div>
27        <div class="item"></div>
28      </div>
29    </main>
30  </body>
31 </html>

```

In case justify-content: center is not there: then vertically only it will be in centre not horizontally.



```

1 flex.html > html > head > style > .container
2 <html lang="en">
3 <head>
4   <style>
5     .container{
6       border: 2px solid red;
7       display: flex;
8       height: 100px;
9     /* justify-content: center; */
10    align-items: center;
11  }
12   .item{
13     height: 12px;
14     width: 12px;
15     background-color: blue;
16     margin: 4px;
17     border: 2px solid black;
18   }
19 </style>
20 </head>
21 <body>
22   <main>
23     <div class="container">
24       <div class="item"></div>
25       <div class="item"></div>
26       <div class="item"></div>
27       <div class="item"></div>
28       <div class="item"></div>
29     </div>
30   </main>
31 </body>
32 </html>

```

We can also do justify-content: space-between: as shown below



```

1 flex.html > html > head > style > .container
2 <html lang="en">
3 <head>
4   <style>
5     .container{
6       border: 2px solid red;
7       display: flex;
8       height: 100px;
9     justify-content:space-between;
10    /* align-items: center; */
11  }
12   .item{
13     height: 12px;
14     width: 12px;
15     background-color: blue;
16     margin: 4px;
17     border: 2px solid black;
18   }
19 </style>
20 </head>
21 <body>
22   <main>
23     <div class="container">
24       <div class="item"></div>
25       <div class="item"></div>
26       <div class="item"></div>
27       <div class="item"></div>
28       <div class="item"></div>
29     </div>
30   </main>
31 </body>
32 </html>

```

We can also do justify-content: space-around: as shown below



```

1 flex.html > html > body > main > div.container > div.item
2 <html lang="en">
3 <head>
4   <style>
5     .container{
6       border: 2px solid red;
7       display: flex;
8       height: 100px;
9     justify-content:space-around;
10    /* align-items: center; */
11  }
12   .item{
13     height: 12px;
14     width: 12px;
15     background-color: blue;
16     margin: 4px;
17     border: 2px solid black;
18   }
19 </style>
20 </head>
21 <body>
22   <main>
23     <div class="container">
24       <div class="item"></div>
25       <div class="item"></div>
26       <div class="item"></div>
27       <div class="item"></div>
28       <div class="item"></div>
29     </div>
30   </main>
31 </body>
32 </html>

```

**So, do you know what actually happens when we apply this flex display property?**

Basically, It Make a container a flex container using:

.container {

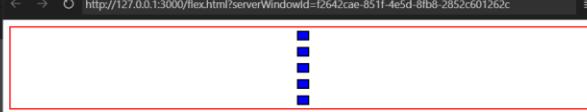
```

display: flex;
}

```

All direct children of .container automatically become flex items.

Now, let's understand the flex-direction:column, basically it will change its direction to column:



The screenshot shows a browser window with the URL <http://127.0.0.1:3000/flex.html?serverWindowId=f2642cae-851f-4e5d-8fb8-2852c601262c>. On the left, the HTML code defines a container with a red border and five items, each with a blue background and black border. On the right, the browser displays five blue squares arranged vertically within a red-bordered container.

```

1 flex.html > html > body > main > div.container > div.item
2   <html lang="en">
3     <head>
4       <style>
5         .container{
6           border: 2px solid red;
7           display: flex;
8           height: 100px;
9           justify-content:space-around;
10          align-items: center;
11          flex-direction: column;
12        }
13        .item{
14           height: 12px;
15           width: 12px;
16           background-color: blue;
17           margin: 4px;
18           border: 2px solid black;
19         }
20       </style>
21     </head>
22   <body>
23     <main>
24       <div class="container">
25         <div class="item"></div>
26         <div class="item"></div>
27         <div class="item"></div>
28         <div class="item"></div>
29         <div class="item"></div>
30       </div>
31     </main>
32   </body>
33 </html>
34
35
36

```

Example: use of flex-direction:column-reverse



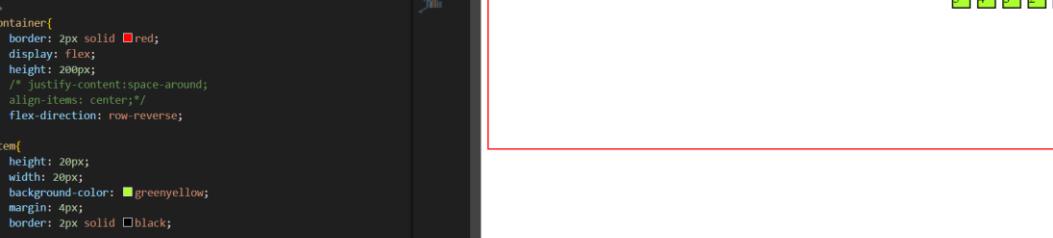
The screenshot shows a browser window with the URL <http://127.0.0.1:3000/flex.html?serverWindowId=f2642cae-851f-4e5d-8fb8-2852c601262c>. On the left, the HTML code defines a container with a red border and five items, each with a greenyellow background and black border. The items are arranged vertically in reverse order (5 at top, 1 at bottom). On the right, the browser displays five green squares arranged vertically within a red-bordered container.

```

1 flex.html > html > body > main > div.container
2   <!DOCTYPE html>
3   <html lang="en">
4     <head>
5       <meta charset="UTF-8">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Document</title>
8       <style>
9         .container{
10           border: 2px solid red;
11           display: flex;
12           height: 200px;
13           justify-content:space-around;
14           align-items: center;
15           flex-direction: column-reverse;
16         }
17         .item{
18           height: 20px;
19           width: 20px;
20           background-color: greenyellow;
21           margin: 4px;
22           border: 2px solid black;
23         }
24       </style>
25     </head>
26   <body>
27     <main>
28       <div class="container">
29         <div class="item">1</div>
30         <div class="item">2</div>
31         <div class="item">3</div>
32         <div class="item">4</div>
33         <div class="item">5</div>
34       </div>
35     </main>
36   </body>
37 </html>
38
39
40
41
42

```

Example: use case of flex-direction:row-reverse



The screenshot shows a browser window displaying a web page. The page content is defined by the following HTML code:

```
flex.html > HTML
1 <html lang="en">
2   <head>
3     <head>
4       <style>
5         .container{
6           border: 2px solid red;
7           display: flex;
8           height: 200px;
9           /* justify-content:space-around;
10          align-items: center; */
11          flex-direction: row-reverse;
12        }
13        .item{
14           height: 20px;
15           width: 20px;
16           background-color: greenyellow;
17           margin: 4px;
18           border: 2px solid black;
19         }
20       </style>
21     </head>
22   <body>
23     <main>
24       <div class="container">
25         <div class="item">1</div>
26         <div class="item">2</div>
27         <div class="item">3</div>
28         <div class="item">4</div>
29         <div class="item">5</div>
30       </div>
31     </main>
32   </body>
33 </html>
```

The browser's status bar at the top right shows the URL `http://127.0.0.1:3000/flex.html?serverWindowId=f2642cae-851f-4e5d-8fb8-2852c601262c`. The main content area displays a red-bordered flex container with five yellow-green square items arranged horizontally from right to left (row-reverse direction). Each item has a black border and a 4px margin.

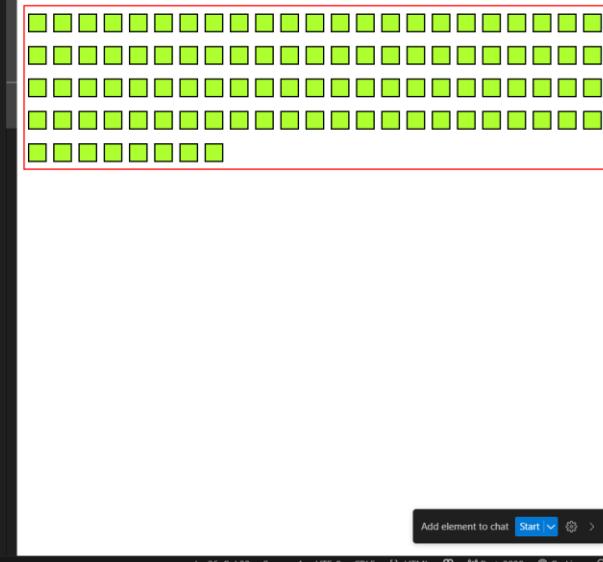
Now, suppose we have increased the number of <div> too much, then it will overflow. As shown below:



The screenshot shows a browser window displaying a web page. The page's source code is visible on the left, and the rendered output is on the right. The source code defines a simple flex layout with a container having 10 items, each with a height and width of 20px and a greenyellow background color. The browser's address bar at the top shows the URL: `http://127.0.0.1:3000/flex.html?serverWindowId=f2642cae-851f-4e5d-8fb8-2852c601262c`.

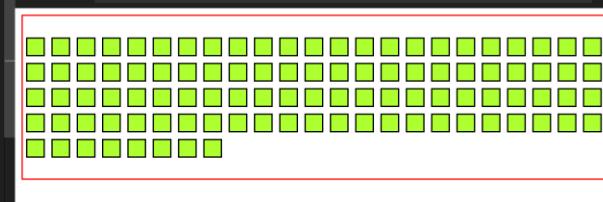
```
flex.html > HTML > body > main > div.container
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7      <style>
8          .container{
9              border: 2px solid red;
10             display: flex;
11             height: 200px;
12             align-items: center;
13         }
14         .item{
15             height: 20px;
16             width: 20px;
17             background-color: greenyellow;
18             margin: 4px;
19             border: 2px solid black;
20         }
21     </style>
22 </head>
23 <body>
24     <main>
25         <div class="container">
26             <div class="item"></div>
27             <div class="item"></div>
28             <div class="item"></div>
29             <div class="item"></div>
30             <div class="item"></div>
31             <div class="item"></div>
32             <div class="item"></div>
33             <div class="item"></div>
34             <div class="item"></div>
35         </div>
36     </main>
37 </body>
38 </html>
```

To fix, this we will use the property, flex-wrap:wrap



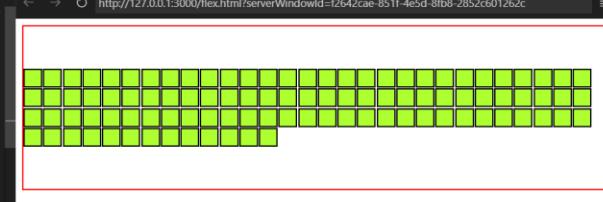
```
flex.html > html > body > main > div.container
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Document</title>
7    <style>
8      .container{
9        border: 2px solid red;
10         display: flex;
11         height: 200px;
12         align-items: center;
13         flex-wrap: wrap;
14     }
15     .item{
16       height: 20px;
17       width: 20px;
18       background-color: greenyellow;
19       margin: 4px;
20       border: 2px solid black;
21     }
22   </style>
23 </head>
24 <body>
25   <main>
26     <div class="container">
27       <div class="item"></div>
28       <div class="item"></div>
29       <div class="item"></div>
30       <div class="item"></div>
31       <div class="item"></div>
32       <div class="item"></div>
33       <div class="item"></div>
34       <div class="item"></div>
35       <div class="item"></div>
36       <div class="item"></div>
37     </div>
38   </main>
39 </body>
40 </html>
```

Example: use case of align-content:center. Basically, we use it when the items be in more than one line because of flex-wrap.



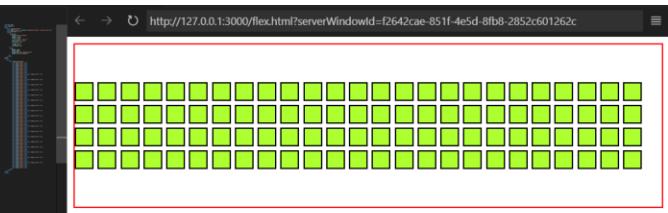
```
flex.html > html > head > style > .item
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Document</title>
7    <style>
8      .container{
9        border: 2px solid red;
10         display: flex;
11         height: 200px;
12         /* align-items: center; */
13         align-content: center;
14         flex-wrap: wrap;
15     }
16     .item{
17       height: 20px;
18       width: 20px;
19       background-color: greenyellow;
20       margin: 4px;
21       border: 2px solid black;
22     }
23   </style>
24 </head>
25 <body>
26   <main>
27     <div class="container">
28       <div class="item"></div>
29       <div class="item"></div>
30       <div class="item"></div>
31       <div class="item"></div>
32       <div class="item"></div>
33       <div class="item"></div>
34       <div class="item"></div>
35       <div class="item"></div>
36       <div class="item"></div>
37     </div>
38   </main>
39 </body>
40 </html>
```

Now, adding the gap property in the container:



```
flex.html > html > body > main > div.container > div.item
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Document</title>
7    <style>
8      .container{
9        border: 2px solid red;
10         display: flex;
11         height: 200px;
12         /* align-items: center; */
13         align-content: center;
14         flex-wrap: wrap;
15         gap: 1px;
16     }
17     .item{
18       height: 20px;
19       width: 20px;
20       background-color: greenyellow;
21       border: 2px solid black;
22     }
23   </style>
24 </head>
25 <body>
26   <main>
27     <div class="container">
28       <div class="item"></div>
29       <div class="item"></div>
30       <div class="item"></div>
31     </div>
32   </main>
33 </body>
34 </html>
```

We can even, separately tell the row and column gap:



The screenshot shows a browser window displaying a grid of green squares. The grid is 8 columns wide and 6 rows high. The squares are arranged in a wrap layout. A red border highlights the entire grid area. To the left of the browser window, there is a code editor showing the corresponding HTML and CSS code.

```
flex.html > html > body > main > div.container > div.item
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7      <style>
8          .container{
9              border: 2px solid red;
10             display: flex;
11             height: 200px;
12             /* align-items: center; */
13             align-content: center;
14             flex-wrap: wrap;
15             /* gap: 1px; */
16             row-gap: 5px;
17             column-gap: 5px;
18         }
19         .item{
20             height: 20px;
21             width: 20px;
22             background-color: greenyellow;
23             border: 1px solid black;
24         }
25     </style>
26  </head>
27  <body>
28      <main>
29          <div class="container">
30              <div class="item"></div>
31              <div class="item"></div>
```

--The End--