Lesson:

Grid item properties

PW SKILLS

Topics Covered

1. Introduction
2. grid-column-start and grid-column-end
3. grid-column
4. grid-row-start and grid-row-end
5. grid-row
6. grid-area
7. justify-self and align-self
8. order

Introduction

Grid item properties are CSS properties that are used to define the layout and positioning of grid items inside a grid container. These properties allow you to control the size, position, and order of grid items within the grid layout.

Now let's discuss some of its main properties.

grid-column-start and grid-column-end

grid-column-start and grid-column-end properties define the starting and ending positions of a grid item along the block axis of the grid respectively. The value can be a line number, a keyword like span, or auto.

index.html

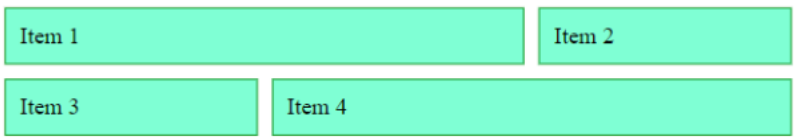
```
<div class="container">
  <div class="item item1">Item 1</div>
  <div class="item item2">Item 2</div>
  <div class="item item3">Item 3</div>
  <div class="item item4">Item 4</div>
</div>
```

style.css

```
.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 10px;
}

.item {
    padding: 10px;
    background-color: aquamarine;
    border: 1px solid green;
}
```

```
.item1 {
    grid-column-start: 1;
    grid-column-end: 3;
}

.item2 {
    grid-column-start: 3;
    grid-column-end: 4;
}

.item3 {
    grid-column-start: 1;
    grid-column-end: 2;
}

.item4 {
    grid-column-start: 2;
    grid-column-end: 4;
}
```

Browser Output



In this example, we have a container with three columns and a gap of 10 pixels. We have four grid items with different grid-column-start and grid-column-end values:

1. **.item1** starts at column line 1 and ends at column line 3.
2. **.item2** starts at column line 3 and ends at column line 4.
3. **.item3** starts at column line 1 and ends at column line 2.
4. **.item4** starts at column line 2 and ends at column line 4.

grid-column

The shorthand grid-column property specifies both the starting and ending positions in a single declaration.

style.css

```
.item {
    padding: 10px;
    background-color: aquamarine;
    border: 1px solid green;
}

.item1 {
  grid-column: 1 / 3;
}

.item2 {
  grid-column: 3 / 4;
}

.item3 {
  grid-column: 1 / 2;
}

.item4 {
  grid-column: 2 / 4;
}
```

The above code produces the same result as we saw while using the grid-column-start and grid-column-end properties,

We can just write grid-column property and give the start and end value separated by a slash (/) as you can see in the above code example

grid-row-start and grid-row-end

Similarly, grid-row-start and grid-row-end properties define the starting and ending positions of a grid item along the vertical axis of the grid.

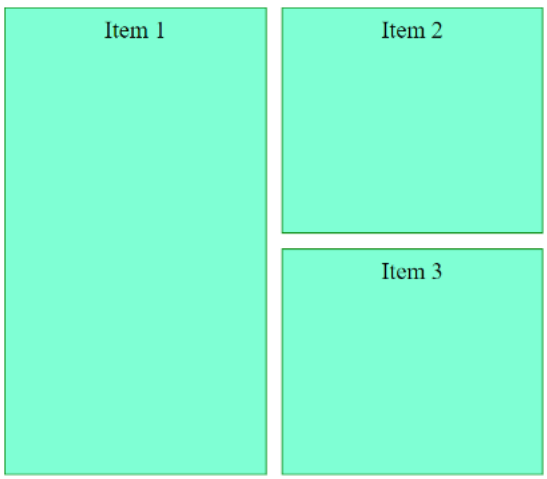
index.html

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

style.css

```
.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(2, 150px);
    grid-gap: 10px;
}

.item {
    padding: 5px;
    background-color: aquamarine;
    border: 1px solid green;
    text-align: center;
}

.item-1 {
    grid-row-start: 1;
    grid-row-end: 3;
}

.item-2 {
    grid-row-start: 1;
    grid-row-end: 2;
}
.item-3 {
  grid-row-start: 2;
  grid-row-end: 3;
}
```

Browser Output



We have three grid items with different grid-row-start and grid-row-end values:

1. **.item1** starts at row line 1 and ends at row line 3.
2. **.item2** starts at row line 1 and ends at row line 2.
3. **.item3** starts at row line 2 and ends at row line 3.

As you can see in the above output, Item 1 starts from row line 1 and ends at row line 3, by mentioning the grid-row-start as 1 and grid-row-end as 3.

grid-row

The shorthand grid-row property to specify both the starting and ending positions in a single declaration. The value can be a line number, a keyword like span, or auto.

style.css

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(2, 150px);
  grid-gap: 10px;
}

.item {
  background-color: #ddd;
  padding: 10px;
  text-align: center;
}

.item-1 {
  grid-row: 1 / 3;
}

.item-2 {
  grid-row: 1 / 2;
}

.item-3 {
  grid-row: 2 / 3;
}

.item-4 {
  grid-row: 2 / 4;
}

.item-5 {
  grid-row: 1 / 3;
}

.item-6 {
  grid-row: 1 / 2;
}
```

This also produces the same output as we got while we doing with grid-row-start and grid-row-end properties, Here the difference is just that we can define the start and end for grid-row in the same line just by separating with a slash (/)

grid-area

This property is a shorthand for all four, grid-row-start, grid-row-end, grid-column-start, grid-column-end.

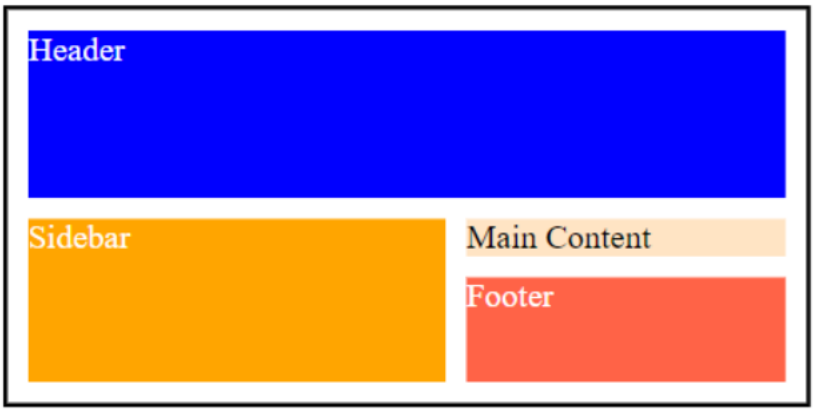
index.html

```
<div class="grid-container">
  <div class="header">Header</div>
  <div class="sidebar">Sidebar</div>
  <div class="main">Main Content</div>
  <div class="footer">Footer</div>
</div>
```

style.css

```
.grid-container {
   display: grid;
   grid-template-areas:
     "header header header"
     "sidebar main main"
     "sidebar footer footer";
   grid-template-rows: 80px 1fr 50px;
   grid-template-columns: 200px 1fr 1fr;
   grid-gap: 10px;
   background-color: #fff;
   border: 2px solid #000;
   padding: 10px;
}

.header {
   grid-area: header;
   background-color: blue;
   color: white;
}

.sidebar {
   grid-area: sidebar;
   background-color: orange;
   color: white;
}

.main {
   grid-area: main;
   background-color: bisque;
}

.footer {
      grid-area: footer;
      background-color: tomato;
      color: white;
}
```

Browser Output



In the example above, we have created a grid container with four child elements, each representing a grid area (header, sidebar, main, footer). The grid-template-areas property defines the grid areas in a grid container, and the grid-template-rows and grid-template-columns properties define the size and placement of the rows and columns respectively. The grid-gap property sets the gap between each grid item.

To style each grid area, we use the grid-area property to assign each child element to a specific grid area.

justify-self and align-self

justify-self and align-self properties control the alignment of any specific grid item within its grid cell along the horizontal and vertical axes, respectively. The values can be a start, end, center, stretch.

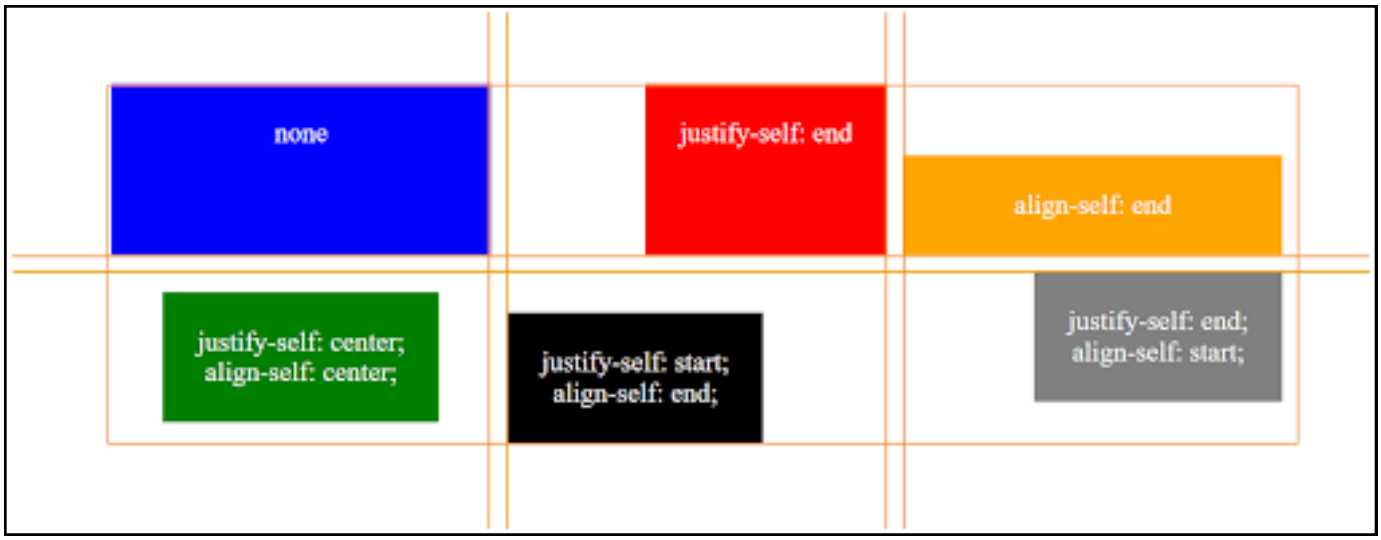
index.html

```
<div class="grid-container">
      <div class="grid-item bg-blue">none</div>
      <div class="grid-item bg-red">justify-self:   end</div>
      <div class="grid-item bg-orange">align-self: end</div>
      <div class="grid-item bg-green">
        justify-self: center; <br />
        align-self: center;
      </div>
      <div class="grid-item bg-black">
        justify-self: start; <br />
        align-self: end;
      </div>
      <div class="grid-item bg-gray">
        justify-self: end; <br />
        align-self: start;
      </div>
</div
```

style.css

```
.grid-container {
  color: white;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(2, 100px);
  grid-gap: 10px;
}
.grid-item {
  background-color: #ddd;
  padding: 20px;
  text-align: center;
}
.grid-item:nth-child(2) {
  justify-self: end;
}
.grid-item:nth-child(3) {
  align-self: end;
}
.grid-item:nth-child(4) {
  justify-self: center;
  align-self: center;
}
.grid-item:nth-child(5) {
  justify-self: start;
  align-self: end;
}
.grid-item:nth-child(6) {
  justify-self: end;
  align-self: start;
}

.bg-blue {
  background-color: blue;
}

.bg-red {
  background-color: red;
}
.bg-orange {
  background-color: orange;
}
.bg-green {
  background-color: green;
}
.bg-black {
  background-color: black;
}
.bg-gray {
  background-color: gray;
}
```

Browser Output



In this example, we have created a grid container with 6 grid items inside. We have set the grid-template-columns property to create 3 columns with equal width, and grid-template-rows property to create 2 rows with a fixed height of 100px. We have also set a grid-gap of 10px to create some space between the grid items.

We have then used the justify-self and align-self properties to position the grid items within their respective grid cells.

For example, we have set the second grid item to align to the end of the cell horizontally with justify-self: end, and the third grid item to align to the bottom of the cell vertically with align-self: end.

The fourth grid item is centered both horizontally and vertically within its cell with justify-self: center and align-self: center.

The fifth and sixth grid items are positioned at the start and end of their respective cells, both horizontally and vertically, using justify-self and align-self together.

order

order property defines the order in which a grid item is displayed within its grid container. The value can be a numeric value or the keyword auto. Lower values come first.

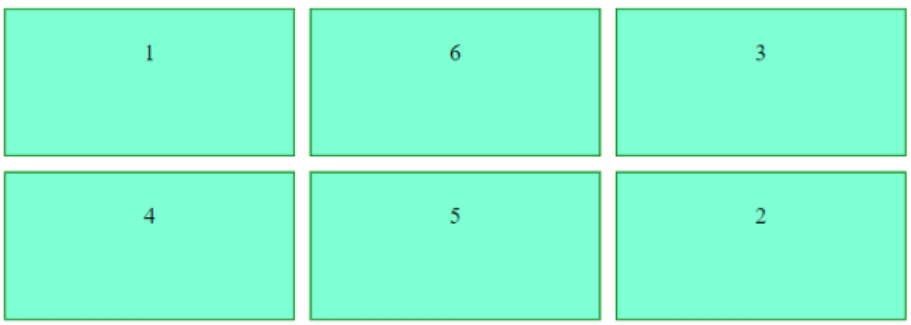
index.html

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
</div>
```

style.css

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(2, 100px);
  grid-gap: 10px;
}

.grid-item {
  background-color: #ddd;
  padding: 20px;
  text-align: center;
}

.grid-item:nth-child(2) {
  order: 6;
}

.grid-item:nth-child(3) {
  order: 2;
}

.grid-item:nth-child(4) {
  order: 3;
}

.grid-item:nth-child(5) {
  order: 5;
}

.grid-item:nth-child(6) {
  order: 1;
}
```

Browser Output



In this example, we have used the order property to specify the order in which the grid items should appear in the grid. The default order value is 0, so we can use positive or negative numbers to reorder the grid items as desired.

For example, we have set the second grid item to have an order of 6, which means it will appear last in the grid. The sixth grid item, on the other hand, has an order of 1, so it will appear first.

The third, fourth, and fifth grid items have been reordered with order values of 2, 3, and 5 respectively, to create a non-sequential ordering of the grid items.