

Grid

The Most Powerful Grid System



SoftUni Team

Technical Trainers



SoftUni



Software University

<https://softuni.org>

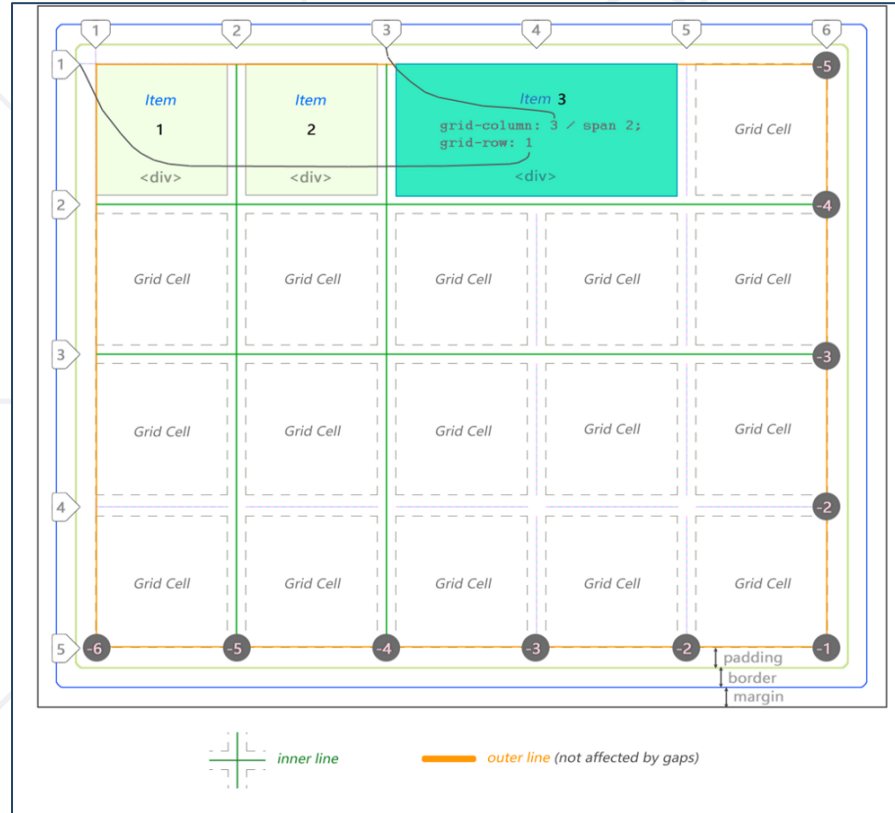
- Grid Layout
- Grid Properties
 - Properties for the Parent (**Grid Container**)
 - Properties for the Children (**Grid Items**)





sli.do

#front-end



Grid


2-dimensional layout system

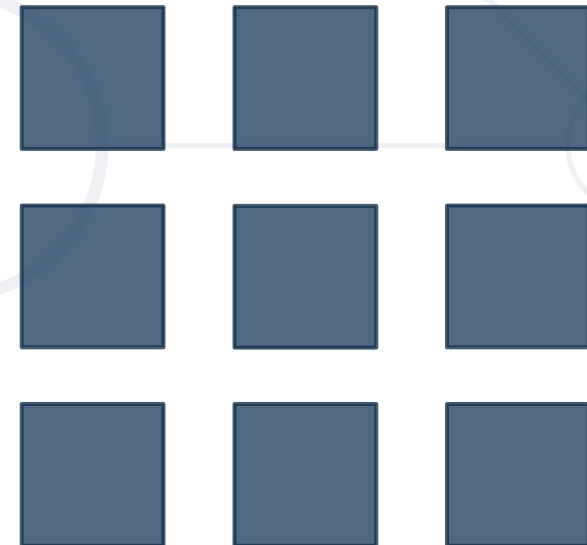
CSS Grid Layout

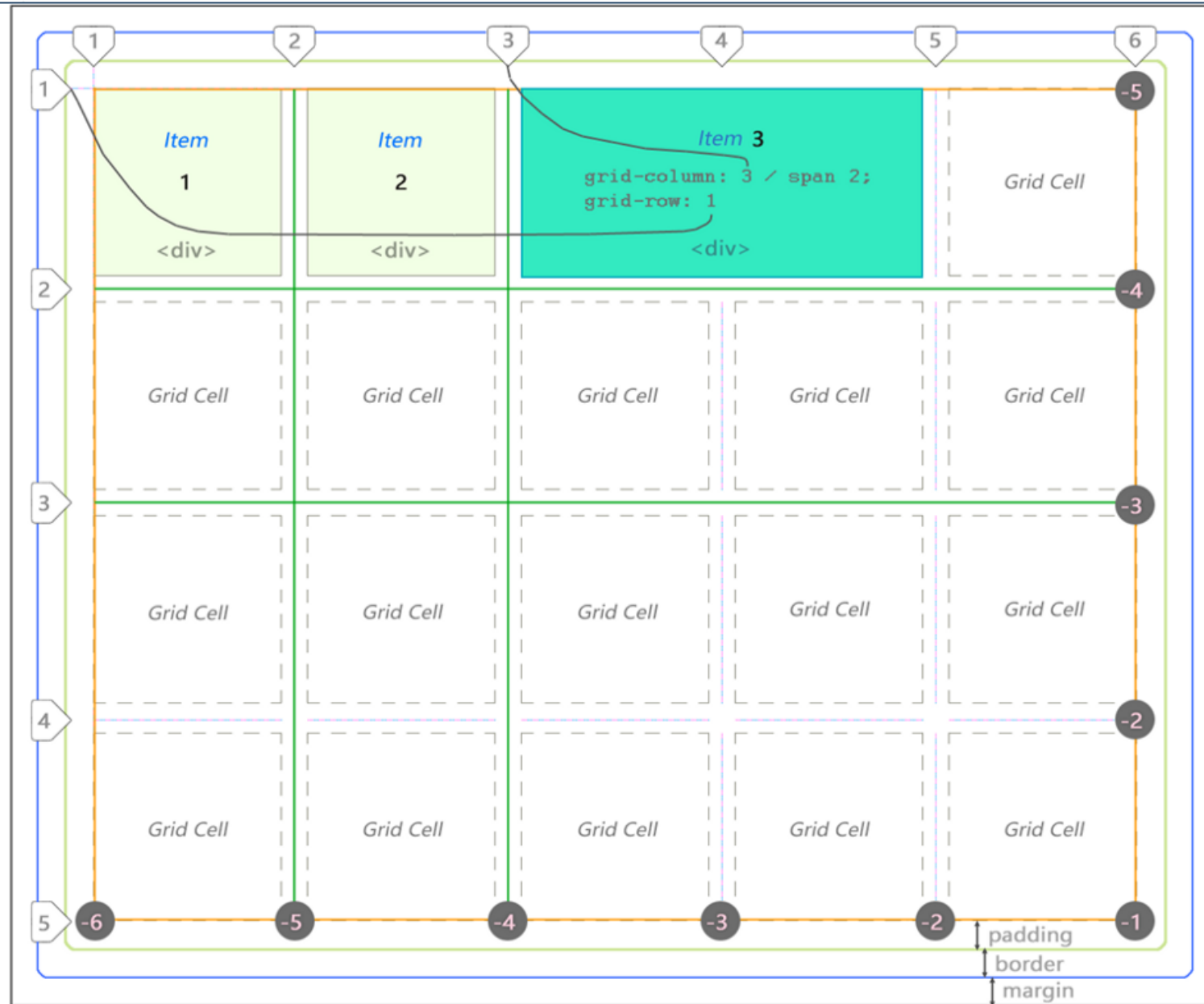


- Excels at **dividing a page** into major regions or defining the relationship in terms of **size**, **position**, and **layer**
- Grid layout enables an author to align elements into **columns** and **rows**

CSS Grid Layout

- 
- Excels at **dividing a page** into major regions or defining the relationship in terms of **size**, **position**, and **layer**
 - Grid layout enables an author to align elements into **columns** and **rows**



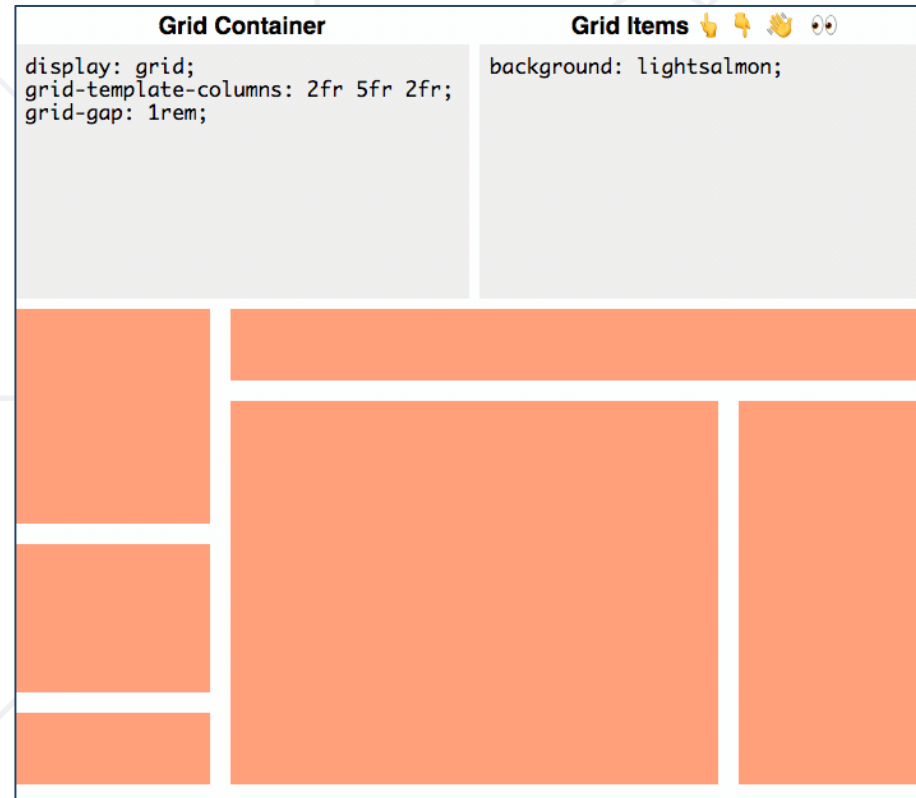


inner line

outer line (not affected by gaps)

padding
border
margin





Grid Properties

Properties for the Parent (Grid Container)

Display

- Defines the element as a grid container and establishes a new grid formatting context for its contents
- Values
 - **grid** - generates a **block-level grid**
 - **inline-grid** - generates an **inline-level grid**



```
.container {  
  display: grid | inline-grid;  
}
```

Grid Column

- A vertical track in a CSS Grid Layout, and is the **space** between two vertical grid lines
- It is defined by the **grid-template-columns** property

```
.grid-container {  
  display: grid;  
  grid-template-columns: auto auto auto auto;  
}
```



Grid Row

- A grid row is a **horizontal track** in a CSS Grid Layout, that is the **space** between two horizontal grid lines
 - It is defined by the **grid-template-rows** property

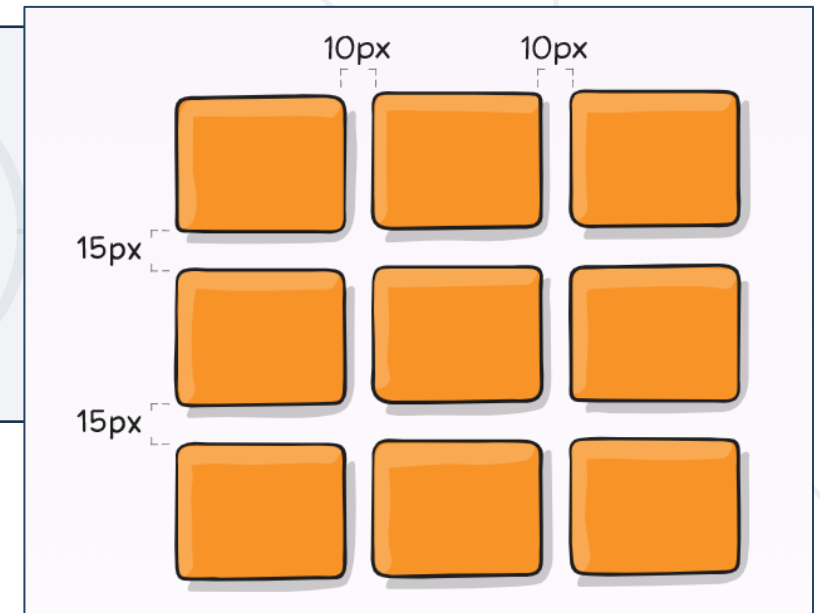
```
.grid-container {  
  display: grid;  
  grid-template-rows: 25% 100px auto;  
}
```



Gutters (Gaps)

- Gutters or alleys are **spacing** between rows and columns
- These can be created using the **grid-column-gap**, **grid-row-gap**, or **grid-gap** properties

```
.container {  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: 80px auto 80px;  
  column-gap: 10px;  
  row-gap: 15px;  
}
```



Grid Units

- The **fr** unit allows you to set the **size of a track** as a fraction of the **free space** of the grid container
 - For example, this will set each item to one third the width of the grid container:

```
.container {  
  grid-template-columns: 1fr 1fr 1fr;  
}
```

- The free space is calculated **after** any non-flexible items



Grid Template Areas

- Defines a **grid template** by referencing the **names** of the grid areas which are specified with the **grid-area** property
 - Repeating the **name** of a **grid area** causes the content to span those cells
 - A period signifies an **empty cell**
 - The syntax itself provides a **visualization** of the structure of the grid

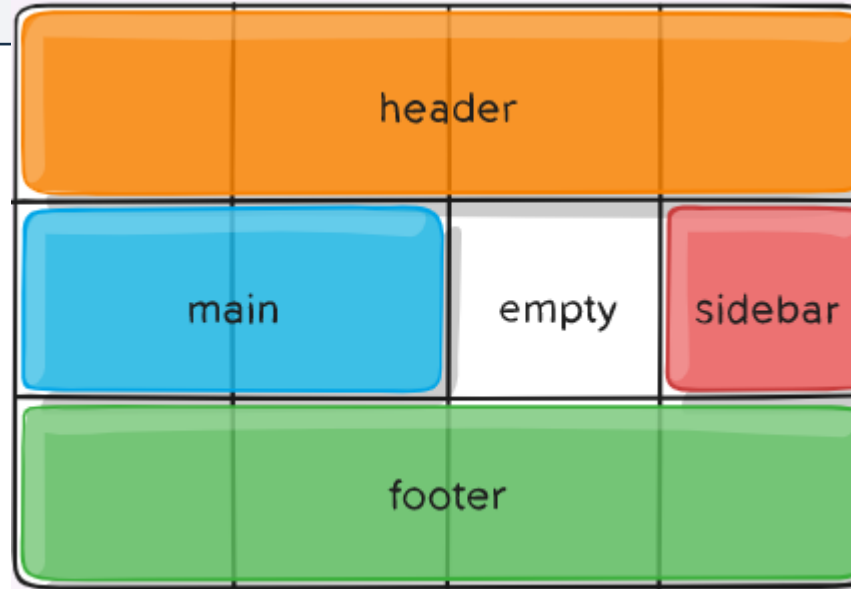


- Values:
 - **<grid-area-name>** - the name of a grid area specified with **grid-area**
 - **.** - a period signifies an empty grid cell
 - **none** - no grid areas are defined

Grid Template Areas - Example

```
.item-a {  
  grid-area: header;  
}  
  
.item-b {  
  grid-area: main;  
}  
  
.item-c {  
  grid-area: sidebar;  
}  
  
.item-d {  
  grid-area: footer;  
}
```

```
.container {  
  display: grid;  
  grid-template-columns: 50px 50px 50px 50px;  
  grid-template-rows: auto;  
  grid-template-areas:  
    'header header header header'  
    'main main . sidebar'  
    'footer footer footer footer';  
}
```



Justify Items

- Aligns grid items along the **inline (row)** axis
- This value applies to **all grid items** inside the container
- Values:
 - **start** - aligns items to be flush with the start edge of their cell
 - **end** - aligns items to be flush with the end edge of their cell
 - **center** - aligns items in the center of their cell
 - **stretch** - fills the whole width of the cell (this is the default)

```
justify-items: start | end | center | stretch;
```



Justify Items - Example

`justify-items: start;`

<div></div>	<div></div>	<div></div>
<div></div>	<div></div>	<div></div>
<div></div>	<div></div>	<div></div>

`justify-items: end;`

<div></div>	<div></div>	<div></div>
<div></div>	<div></div>	<div></div>
<div></div>	<div></div>	<div></div>

`justify-items: center;`

<div></div>	<div></div>	<div></div>
<div></div>	<div></div>	<div></div>
<div></div>	<div></div>	<div></div>

`justify-items: stretch;`

<div></div>	<div></div>	<div></div>
<div></div>	<div></div>	<div></div>
<div></div>	<div></div>	<div></div>

Align Items

- Aligns grid items along the **block (column)** axis (as opposed to **justify-items** which aligns along the inline (row) axis)
- This value applies to all grid items inside the container
- Values:
 - **start** - aligns items to be flush with the start edge of their cell
 - **end** - aligns items to be flush with the end edge of their cell
 - **center** - aligns items in the center of their cell
 - **stretch** - fills the whole height of the cell (this is the default)

```
align-items: start | end | center | stretch;
```

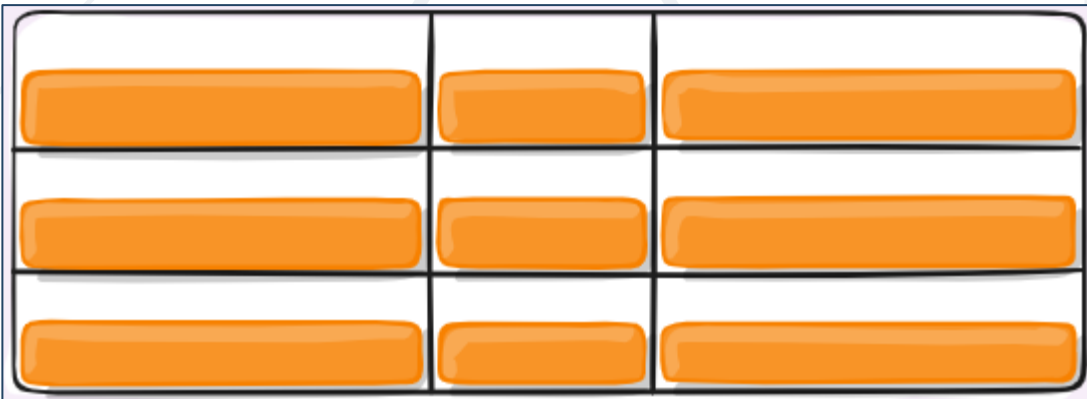


Align Items - Example

`align-items: start;`



`align-items: end;`



`align-items: center;`



`align-items: stretch;`



Justify Content

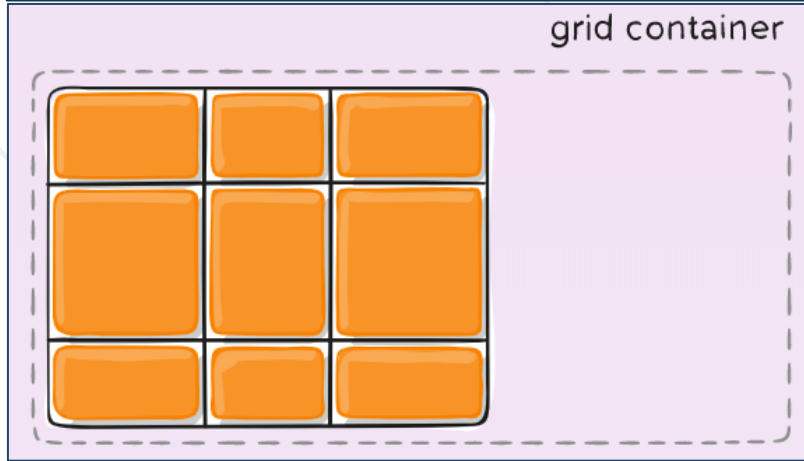
- Sometimes the total size of your grid might be **less** than the size of its grid container
 - This could happen if all of your grid items are sized with **non-flexible units** like px
 - In this case you can set the alignment of the grid within the **grid container**
- Values:

```
justify-content: start | end | center | stretch |  
space-around | space-between | space-evenly;
```

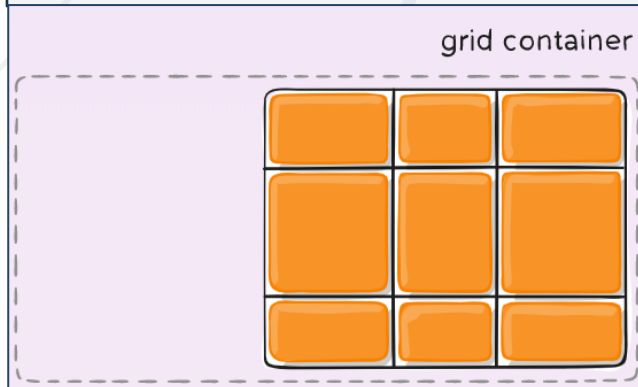


Justify Content - Example

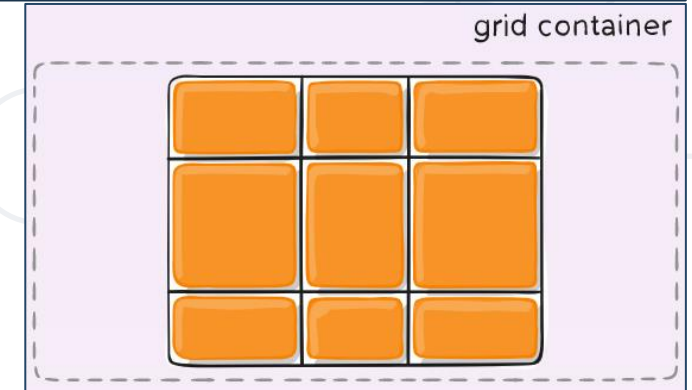
justify-content: start;



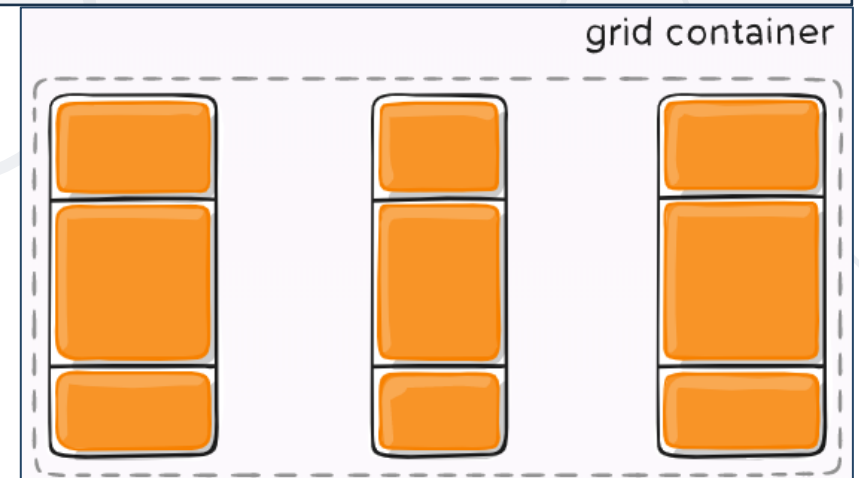
justify-content: end;



justify-content: center;



justify-content: space-between;



Align Content

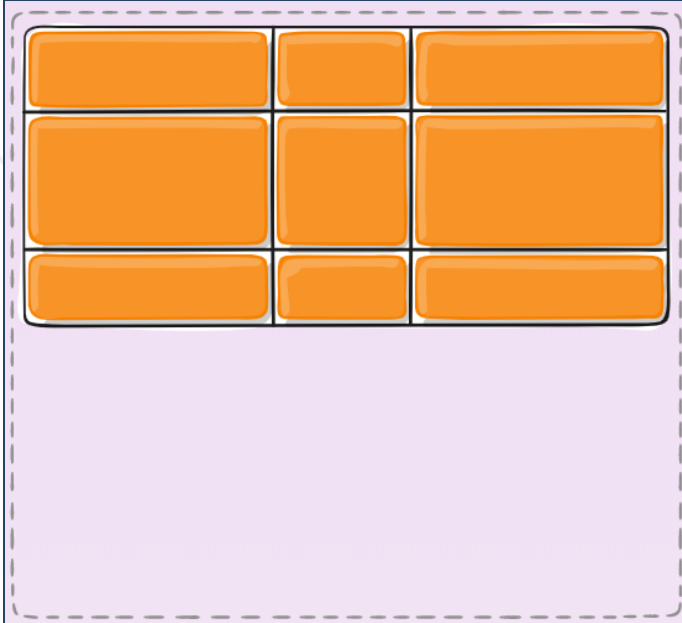
- This property aligns the grid along the **block (column)** axis (as opposed to **justify-content** which aligns the grid along the **inline (row)** axis)

```
.container {  
  align-content: start | end | center | stretch  
  | space-around | space-between | space-evenly;  
}
```

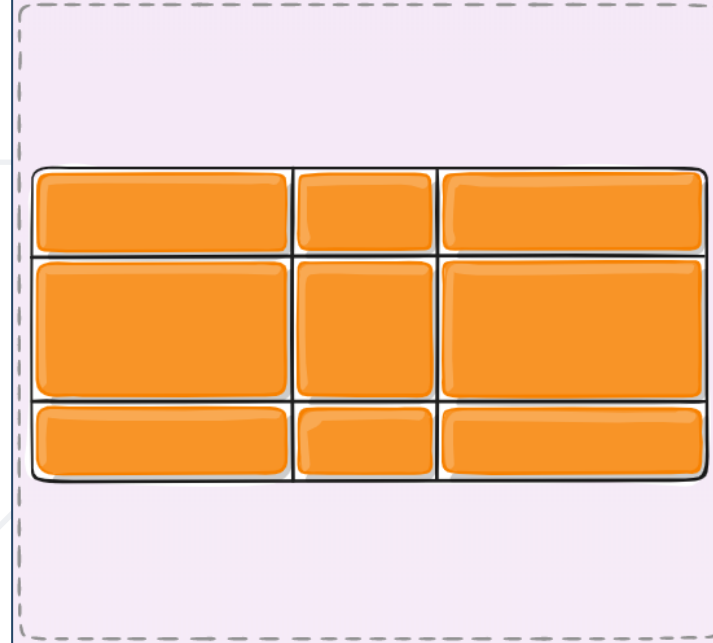


Align Content - Example

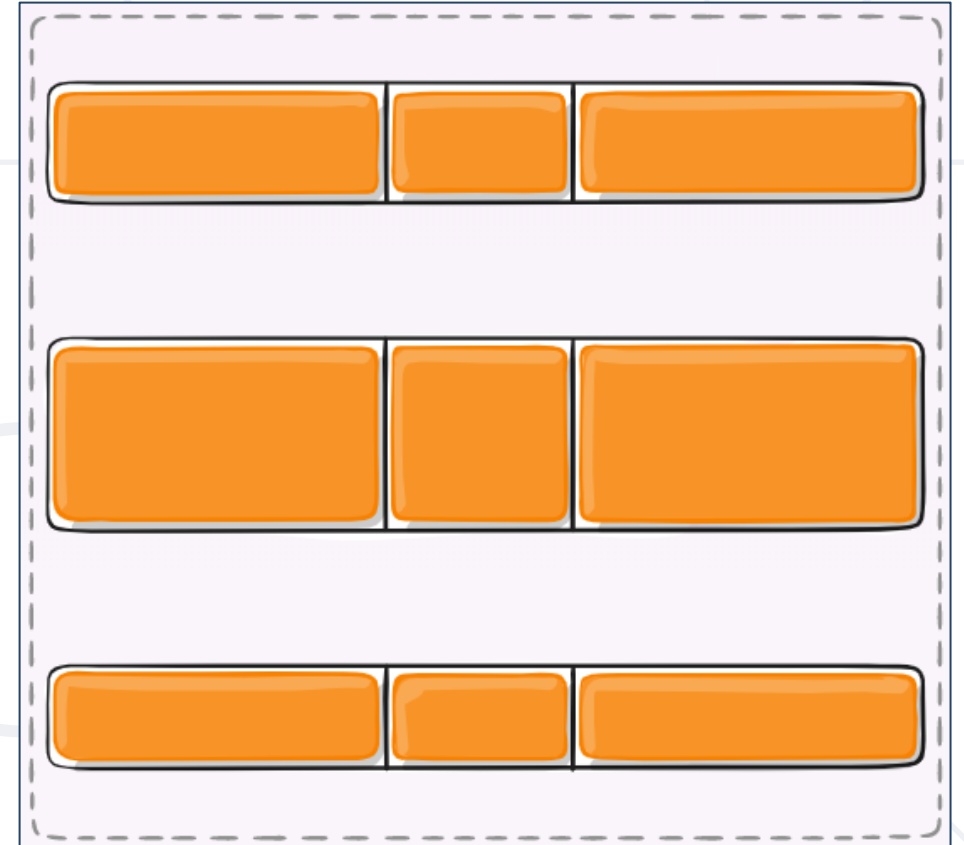
align-content: start;



align-content: center;



align-content: space-around;

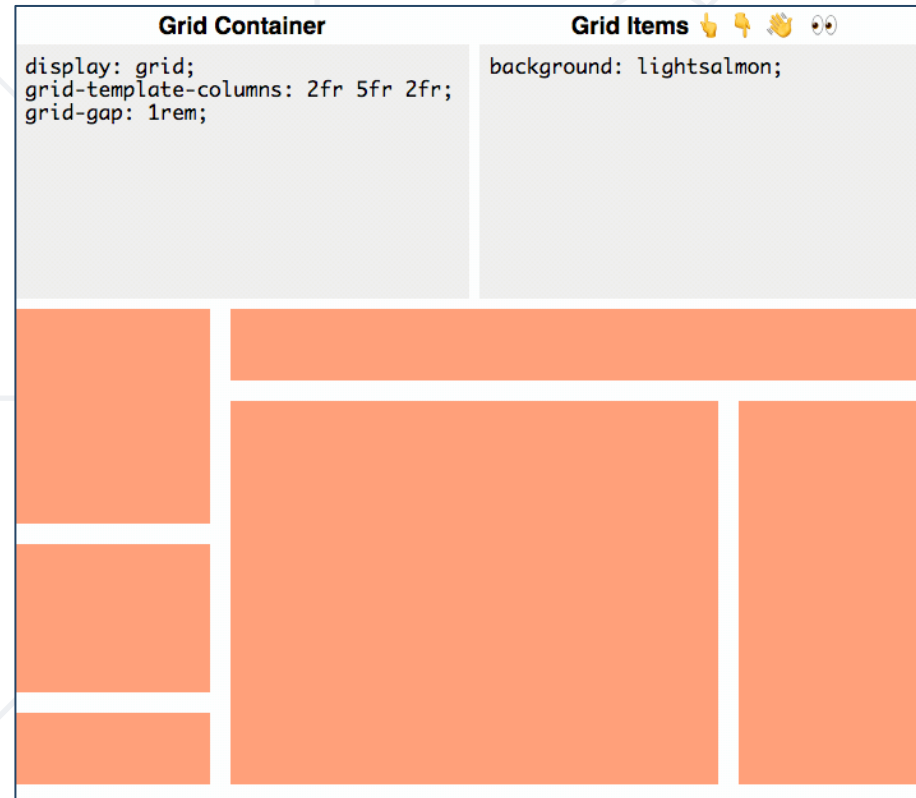


Grid Auto Flow

- If you have grid items that you don't explicitly place on the grid, the **auto-placement algorithm** kicks in to automatically place the items
- This property **controls** how the auto-placement algorithm works

```
.container {  
  grid-auto-flow: row | column | row dense | column dense;  
}
```





Grid Properties

Properties for the Children (Grid Items)

Placing Grid Items

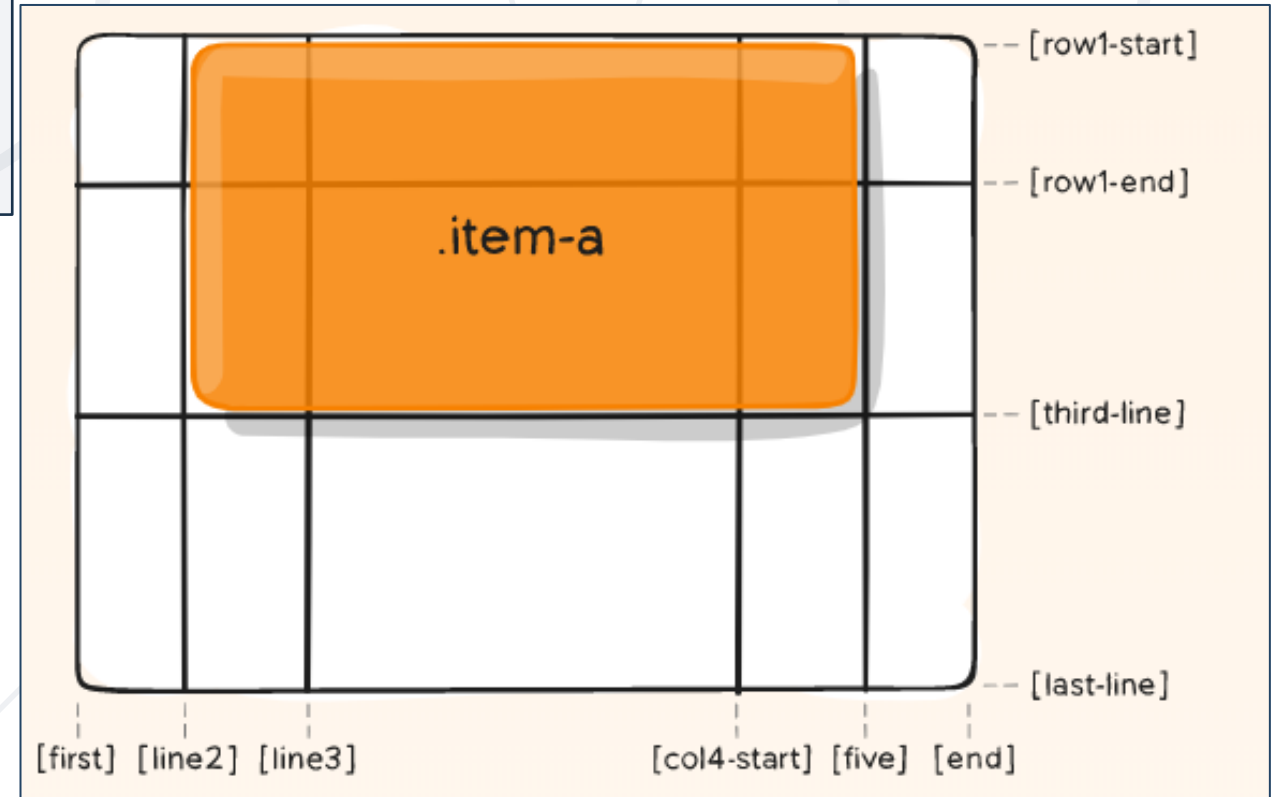
- Determines a **grid item's location** within the grid by referring to specific grid lines
 - **grid-column-start/grid-row-start** is the line where the item **begins**
 - **grid-column-end/grid-row-end** is the line where the item **ends**



```
grid-column-start: <number> | <name> | span <number> | span <name> | auto
grid-column-end: <number> | <name> | span <number> | span <name> | auto
grid-row-start: <number> | <name> | span <number> | span <name> | auto
grid-row-end: <number> | <name> | span <number> | span <name> | auto
```

Placing Grid Items - Example

```
.item-a {  
  grid-column-start: 2;  
  grid-column-end: 5;  
  grid-row-start: 1;  
  grid-row-end: 3;  
}
```



Grid Column, Grid Row

- A shorthand property for **grid-row-start** and **grid-row-end**
- Specifying a **grid item's size** and **location** within the grid row by contributing a **line**, a **span**, or nothing (**automatic**) to its grid placement
- Specifying the **inline-start** and **inline-end** edge of its grid area



```
grid-column: 3 / span 2;  
grid-row: third-line / 4;
```

- Aligns a grid item inside a **cell** along the **inline (row) axis** (as opposed to **align-self** which aligns along the block (column) axis)
- This value applies to a grid item inside a **single cell**

```
.item {  
  justify-self: start | end | center | stretch;  
}
```

Justify Self - Example

```
justify-self: start;
```

.item-a		

```
justify-self: end;
```

	.item-a	

```
justify-self: center;
```

.item-a		

```
justify-self: stretch;
```

.item-a		

- Aligns a grid item **inside** a **cell** along the **block (column)** axis (as opposed to **justify-self** which aligns along the inline (row) axis)
- This value applies to the content inside a **single grid item**

```
.item {  
  align-self: start | end | center | stretch;  
}
```


Align Self - Example

align-self: start;

.item-a		

align-self: end;

.item-a		

align-self: center;

.item-a		

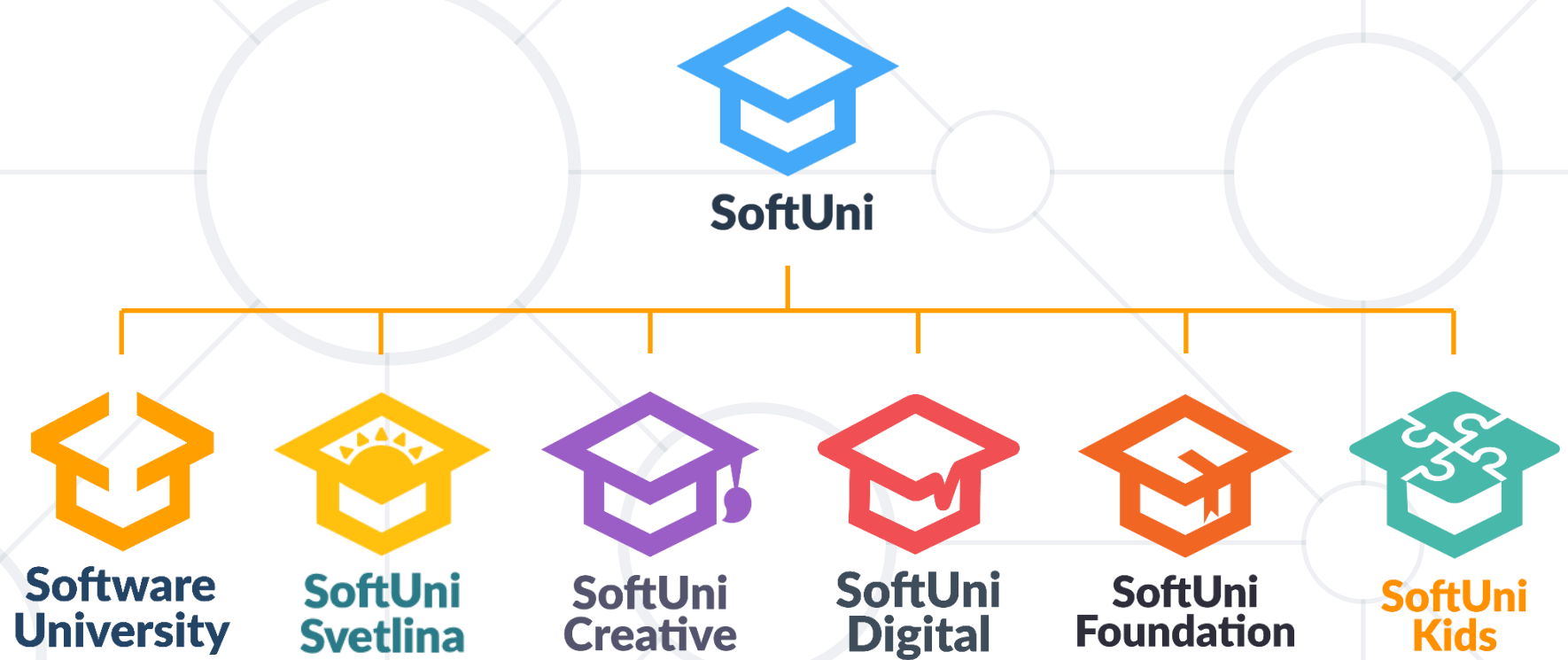
align-self: stretch;

.item-a		

- Grid Layout
- Grid Properties
 - Properties for the Parent (**Grid Container**)
 - Properties for the Children (**Grid Items**)



Questions?



SoftUni Diamond Partners



NETPEAK



Postbank

Решения за твоето утре



SoftwareGroup
doing it right



SmartIT



Coca-Cola HBC
Bulgaria

INDEAVR

Serving the high achievers

tek
experts



SBTech
we know sports



INFRAGISTICS®

SUPERHOSTING.BG



telenor

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni - <https://softuni.org>
- © Software University - <https://softuni.bg>



- Software University - High-Quality Education, Profession and Job for Software Developers
 - softuni.bg
 - Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg

