CSS Box Model Cheatsheet

1. What is the CSS Box Model?

Definition: The CSS box model represents the structure of a web page element. It consists of margins, borders, padding, and the actual content.

2. Components of the Box Model

a. Content

Description: The actual text, image, or media in the element.

Properties:

- width
- height

b. Padding

Description: The space between the content and the border.

Properties:

- padding-top
- padding-right
- padding-bottom
- padding-left
- padding (shorthand)

c. Border

Description: The line surrounding the padding (if any) and content.

Properties:

- border-width
- border-style
- border-color
- border (shorthand)
- Specific sides (e.g., border-top, border-right)

d. Margin

Description: The space outside the border.

Properties:

- margin-top
- margin-right
- margin-bottom

margin-left

margin (shorthand)

3. Box Sizing

a. box-sizing

Description: Determines how the total width and height of an element are calculated.

Values:

content-box (default): width and height apply to the content only.

border-box: width and height include content, padding, and border.

4. Inspecting the Box Model

DevTools: Use browser developer tools (e.g., Chrome DevTools) to inspect and visualize the box model for any element on a webpage.

Margin vs. Padding

Margin

Definition: The margin is the space outside the border of an element.

Purpose: It is used to create space between the element and its neighboring elements, effectively pushing them apart.

Properties:

margin-top

margin-right

margin-bottom

margin-left

margin (shorthand)

Padding

Definition: The padding is the space between the content of an element and its border.

Purpose: It is used to create space inside an element, effectively pushing the content away from the border.

Properties:

padding-top

padding-right

padding-bottom

padding-left

padding (shorthand)

Key Differences

Location:

Margin: Outside the border.

Padding: Inside the border.

Effect on Element Size:

Margin: Does not affect the element's size; it affects the space around the element.

Padding: Increases the element's total size by adding space inside the border, around the content.

Background Color:

Margin: Transparent, so it does not show the element's background color.

Padding: Takes on the element's background color.

Practical Usage :

Margin

To create space between different elements on a page.

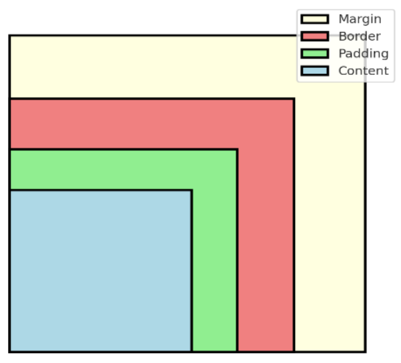
To control the layout and spacing of elements in a design.

Padding

To create space between the content of an element and its border.

To ensure the content is not too close to the border, improving readability and aesthetics.

Understanding the difference between margin and padding and how to use them effectively is essential for creating well-structured and visually appealing web layouts.



CSS Selectors Cheatsheet

CSS selectors are patterns used to select and style the elements on a web page. Here’s a comprehensive guide to CSS selectors, including some advanced tips and tricks.

Basic Selectors

1. Universal Selector (*)

- Description: Selects all elements.
- Example:

```css
* {
  margin: 0;
  padding: 0;
}
```

2. Type Selector

- Description: Selects all elements of a given type.
- Example:

```css
p {
  color: blue;
}
```

3. Class Selector

- Description: Selects all elements with a given class.
- Syntax: .classname
- Example:

```css
.example {
  font-size: 20px;
}
```

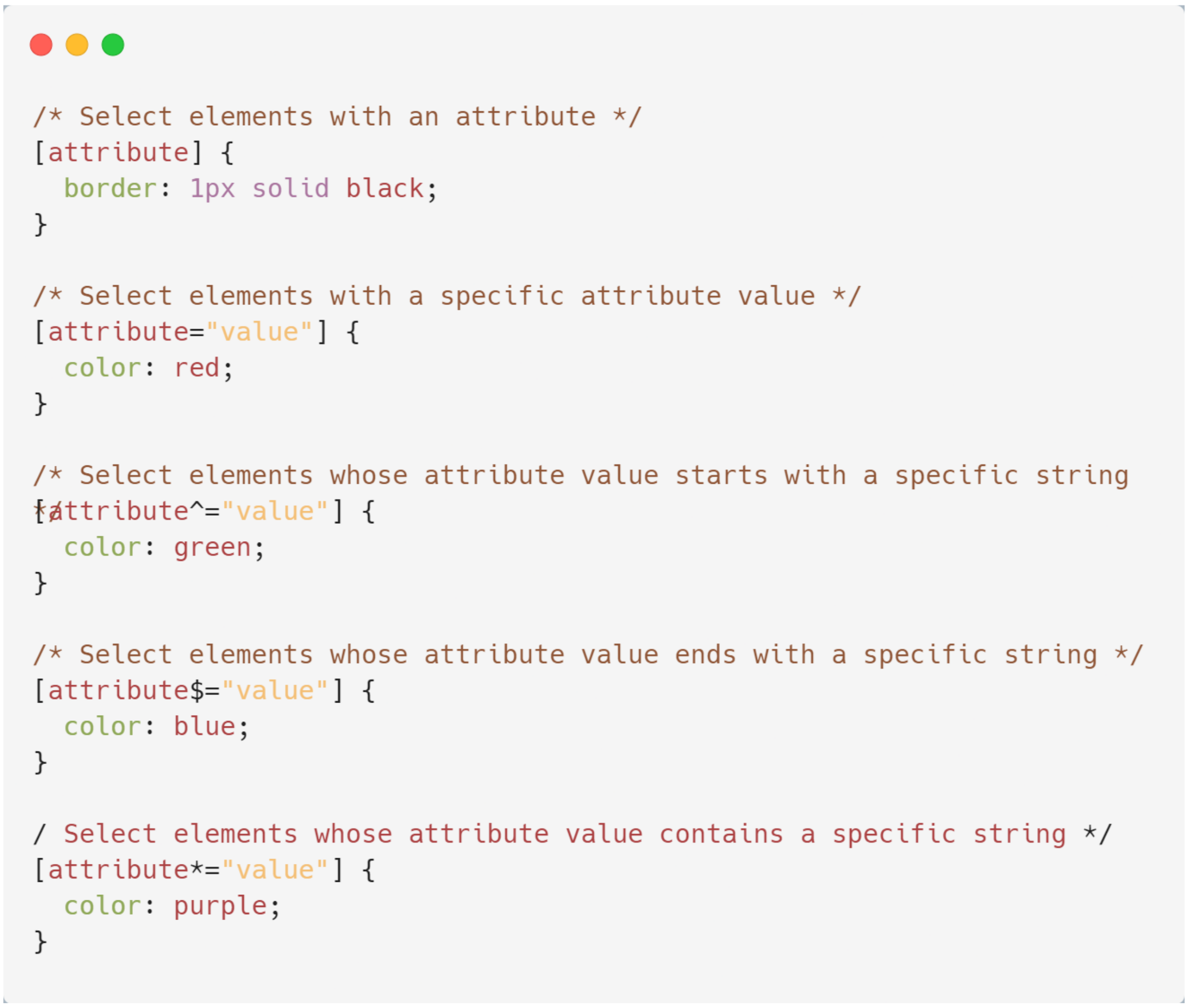
4. ID Selector

- Description: Selects an element with a specific ID.
- Syntax: #idname
- Example:

```css
#unique {
  background-color: yellow;
}
```

5. Attribute Selector

- Description: Selects elements with a specific attribute.
- Examples:



```css
/* Select elements with an attribute */
[attribute] {
  border: 1px solid black;
}

/* Select elements with a specific attribute value */
[attribute="value"] {
  color: red;
}

/* Select elements whose attribute value starts with a specific string
*/
[attribute^="value"] {
  color: green;
}

/* Select elements whose attribute value ends with a specific string */
[attribute$="value"] {
  color: blue;
}

/ Select elements whose attribute value contains a specific string */
[attribute*="value"] {
  color: purple;
}
```

Combinators

1. Descendant Combinator ( )

- Description: Selects all elements that are descendants of a specified element.
- Example:

```
div p {
  color: brown;
}
```

2. Child Combinator (>)

- Description: Selects all elements that are direct children of a specified element.
- Example:

```
ul > li {
  list-style-type: none;
}
```

3. Adjacent Sibling Combinator (+)

- Description: Selects an element that is directly after another specified element.
- Example:

```
h1 + p {
  margin-top: 0;
}
```

4. General Sibling Combinator (~)

- Description: Selects all elements that are siblings of a specified element.
- Example:

```
h1 ~ p {
  color: gray;
}
```

Pseudo-classes

1. :hover

- Description: Selects elements when you mouse over them.
- Example:

```css
a:hover {
  color: orange;
}
```

2. :focus

- Description: Selects elements when they gain focus.
- Example:

```css
input:focus {
  border-color: blue;
}
```

3. :nth-child(n)

- Description: Selects the nth child of a parent element.
- Example:

```css
li:nth-child(2) {
  background-color: lightblue;
}
```

4. :nth-of-type(n)

- Description: Selects the nth child of a parent element, of a specific type.
- Example:

```css
p:nth-of-type(2) {
  font-weight: bold;
}
```

5. :first-child

- Description: Selects the first child of a parent element.
- Example:

```css
p:first-child {
  margin-top: 0;
}
```

6. :last-child

- Description: Selects the last child of a parent element.
- Example:

```css
p:last-child {
  margin-bottom: 0;
}
```

7. :not(selector)

- Description: Selects elements that do not match the specified selector.
- Example:

```css
p:not(.intro) {
  color: green;
}
```

Pseudo-elements

1. ::before

- Description: Inserts content before an element’s content.
- Example:

```css
p::before {
  content: "Note: ";
  font-weight: bold;
}
```

2. ::after

- Description: Inserts content after an element’s content.
- Example:

```css
p::after {
  content: " End.";
  font-weight: bold;
}
```

3. ::first-letter

- Description: Selects the first letter of a block-level element.
- Example:

```css
p::first-letter {
  font-size: 2em;
  color: red;
}
```

4. ::first-line

- Description: Selects the first line of a block-level element.
- Example:

```css
p::first-line {
  font-weight: bold;
}
```

Advanced Tips

1. Chaining Selectors

- Description: Combine multiple selectors to be more specific.
- Example:

```
div.classname#idname[attr="value
"] {
  color: purple;
}
```

2. Combining Pseudo-classes and Pseudo-elements

- Description: Use pseudo-classes and pseudo-elements together for advanced styling.
- Example:

```
div.classname#idname[attr="value
"] {
  color: purple;
}
```

3. Selector Specificity

- Description: Understanding the hierarchy of selectors can help you manage CSS conflicts.
    - Inline styles: 1000
    - ID selectors: 100
    - Class selectors, attributes selectors, and pseudo-classes: 10
    - Type selectors and pseudo-elements: 1
    - Universal selector and combinators: 0

By mastering these CSS selectors, you can target and style any element on your webpage with precision and flexibility. This foundation is crucial for building responsive and visually appealing web designs.

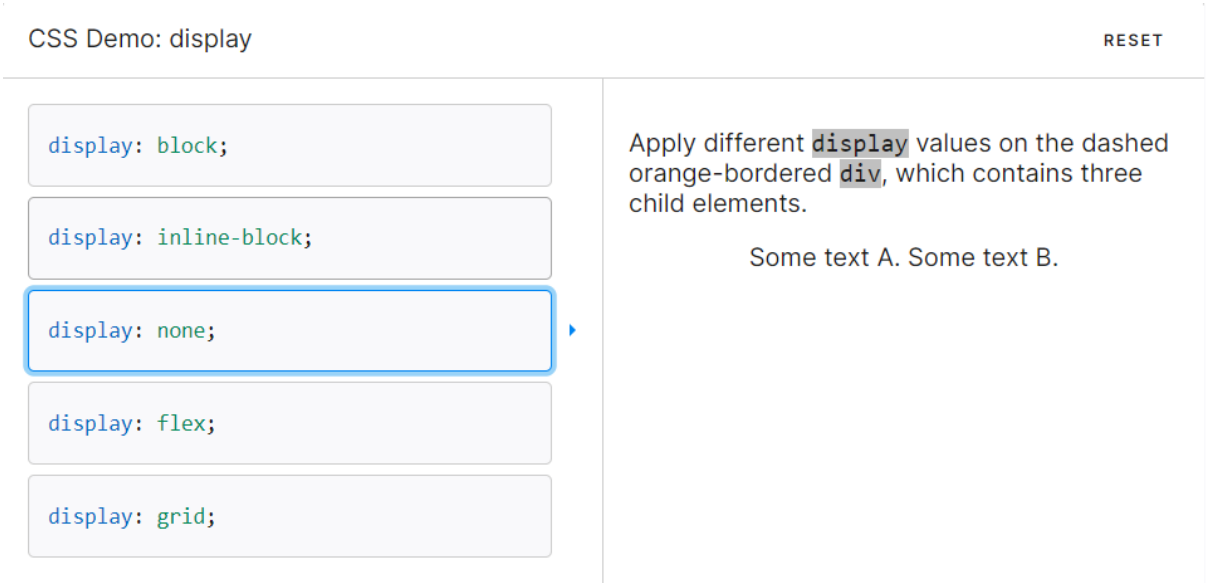
CSS display Property Cheatsheet

The display property in CSS is fundamental for controlling the layout of an element on a web page. It determines how an element is displayed and how it interacts with other elements.

1. Basic Values

display: none

- Description: The element is not displayed at all (it has no effect on layout).
- Use Case: Hiding elements without deleting them from the HTML.



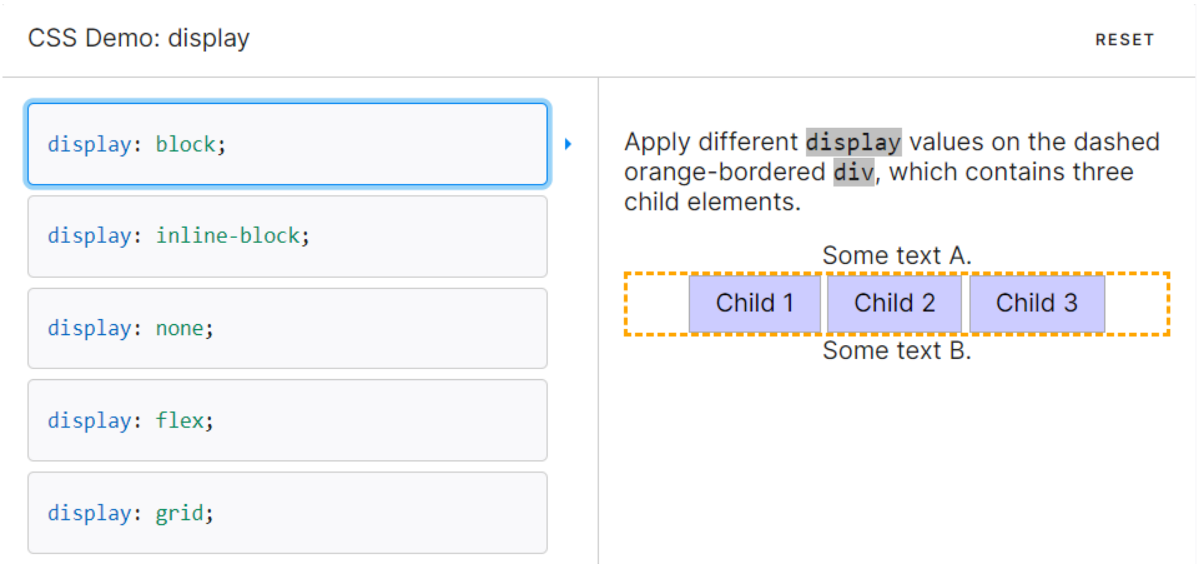
```
display: block;
display: inline-block;
display: none;
display: flex;
display: grid;
```

Apply different `display` values on the dashed orange-bordered `div`, which contains three child elements.

Some text A. Some text B.

display: block

- Description: The element is displayed as a block element (e.g., <div>, <p>).
- Characteristics:
  - Takes up the full width available.
  - Starts on a new line.
- Use Case: Structuring sections of content, such as paragraphs, divs, and headers.



```
display: block;
display: inline-block;
display: none;
display: flex;
display: grid;
```

Apply different `display` values on the dashed orange-bordered `div`, which contains three child elements.

Some text A.

Child 1 Child 2 Child 3

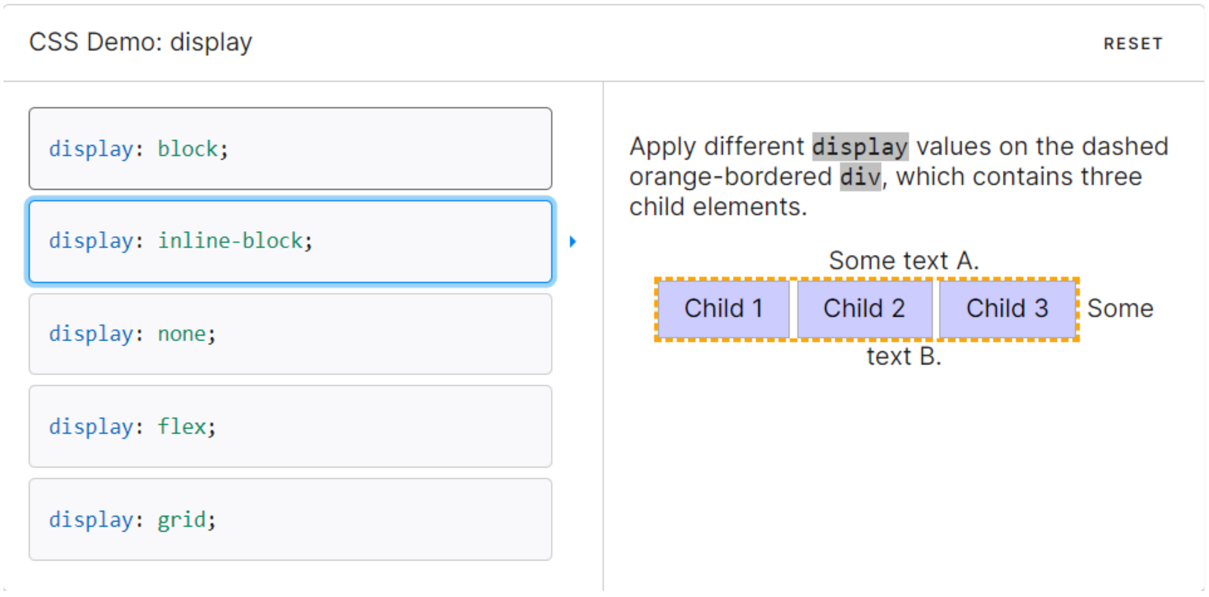
Some text B.

display: inline

- Description: The element is displayed as an inline element (e.g., <span>, <a>).
- Characteristics:
- Takes up only as much width as necessary.
- Does not start on a new line.
- Use Case: Styling text within a paragraph, links, or small pieces of content within a line.
- **inline**
    - The element generates one or more inline boxes that do not generate line breaks before or after themselves. In normal flow, the next element will be on the same line if there is space.

display: inline-block

- Description: Combines features of both inline and block elements.
- Characteristics:
- Takes up only as much width as necessary (like inline).
- Allows setting width and height (like block).
- Use Case: Creating buttons, styled links, or aligning multiple items horizontally.



```
display: block;
display: inline-block;
display: none;
display: flex;
display: grid;
```

Apply different `display` values on the dashed orange-bordered `div`, which contains three child elements.

Flexbox Guide

Overview

Flexbox, or the Flexible Box Layout, is a layout model that allows elements within a container to be automatically arranged depending on screen size, making it perfect for responsive design. It aligns items efficiently, both horizontally and vertically.

Key Concepts

Flex Container and Flex Items

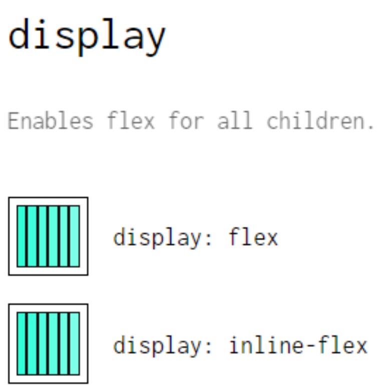
- ○ Flex Container: The parent element that has display: flex or display: inline-flex.
- ○ Flex Items: The direct children of a flex container.

Main Properties of Flexbox

Container Properties

display

- flex: Defines a block-level flex container.
- inline-flex: Defines an inline-level flex container.



display

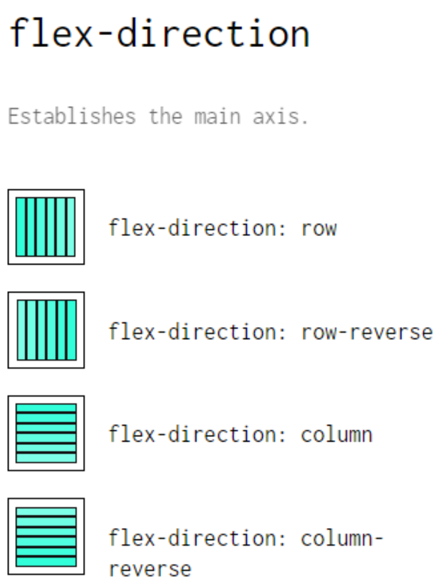
Enables flex for all children.

display: flex

display: inline-flex

flex-direction

- row: Default value; items are placed in a horizontal line.
- row-reverse: Items are placed in a horizontal line but in reverse order.
- column: Items are placed in a vertical line.
- column-reverse: Items are placed in a vertical line but in reverse order.



flex-direction

Establishes the main axis.

flex-direction: row

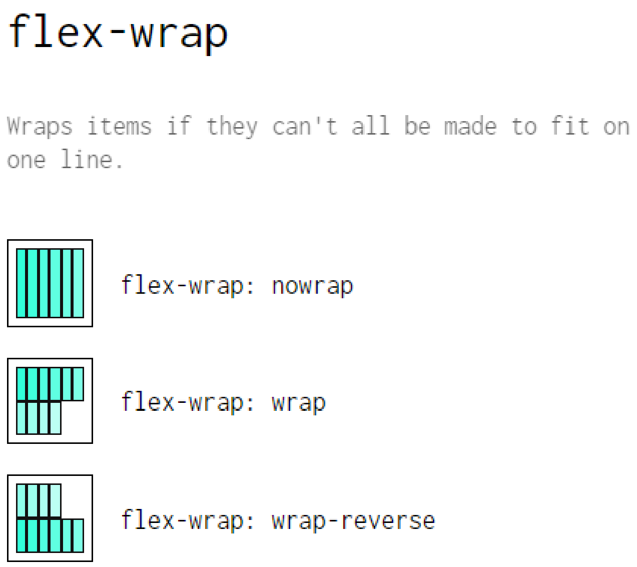
flex-direction: row-reverse

flex-direction: column

flex-direction: column-reverse

flex-wrap

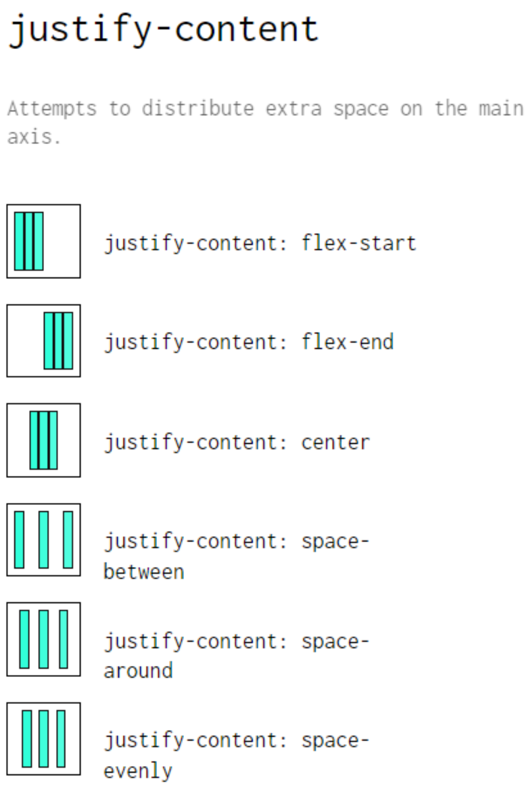
- nowrap: Default value; items will not wrap.
- wrap: Items will wrap onto multiple lines.
- wrap-reverse: Items will wrap onto multiple lines in reverse order


flex-wrap

Wraps items if they can't all be made to fit on one line.

justify-content

- flex-start: Items are packed toward the start of the container.
- flex-end: Items are packed toward the end of the container.
- center: Items are centered along the main axis.
- space-between: Items are evenly distributed; first item is at the start, last item is at the end.
- space-around: Items are evenly distributed with equal space around them.
- space-evenly: Items are distributed so that the spacing between any two items (and the space to the edges) is equal.

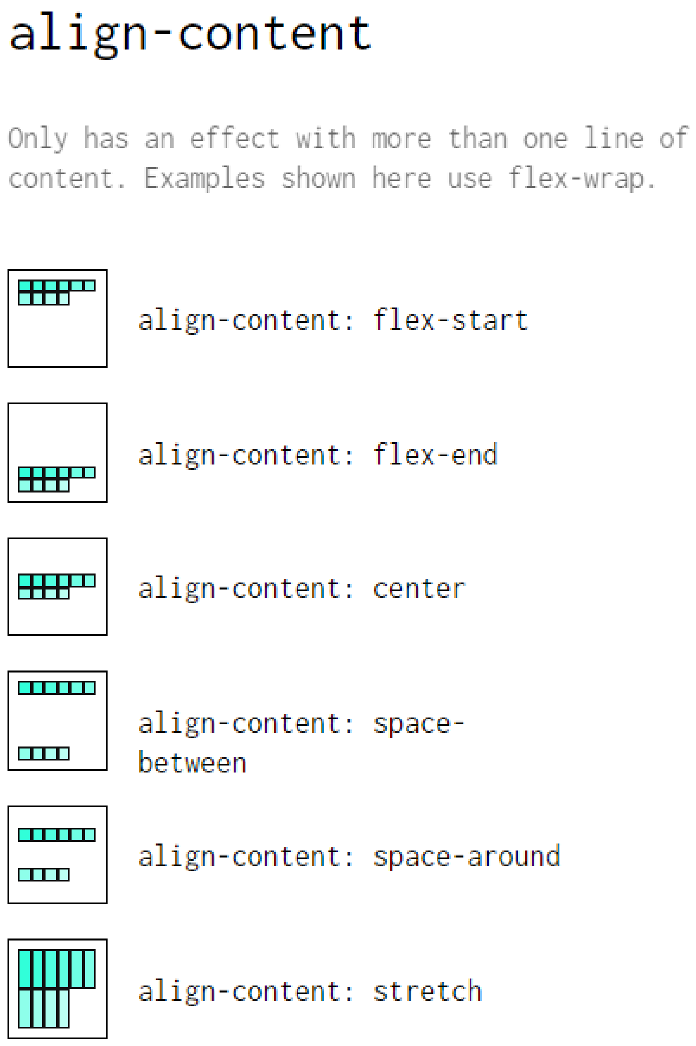

justify-content

Attempts to distribute extra space on the main axis.

justify-content: flex-start
justify-content: flex-end
justify-content: center
justify-content: space-between
justify-content: space-around
justify-content: space-evenly

align-content

- flex-start: Lines are packed toward the start of the flex container.
- flex-end: Lines are packed toward the end of the flex container.
- center: Lines are packed toward the center of the flex container.
- space-between: Lines are evenly distributed with no space between them.
- space-around: Lines are evenly distributed with space around them.
- stretch: Lines stretch to take up the remaining space.



align-content

Only has an effect with more than one line of content. Examples shown here use flex-wrap.

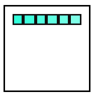
align-content: flex-start
align-content: flex-end
align-content: center
align-content: space-between
align-content: space-around
align-content: stretch

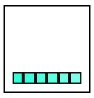
align-items

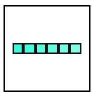
- flex-start: Items are aligned at the start of the cross axis.
- flex-end: Items are aligned at the end of the cross axis.
- center: Items are centered in the cross axis.
- baseline: Items are aligned such as their baselines align.
- stretch: Items are stretched to fit the container.

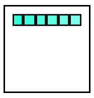
align-items

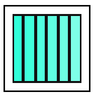
Determines how items are laid out on the cross axis.


align-items: flex-start


align-items: flex-end


align-items: center


align-items: baseline

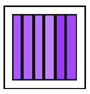

align-items: stretch

Children

order

Defines the order of the flex items. The default is 0. Items with the same order are displayed in the source order.

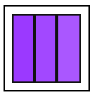
Allows you to explictly set the order you want each child to appear in.

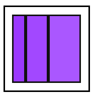

order: integer

flex-grow

Defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

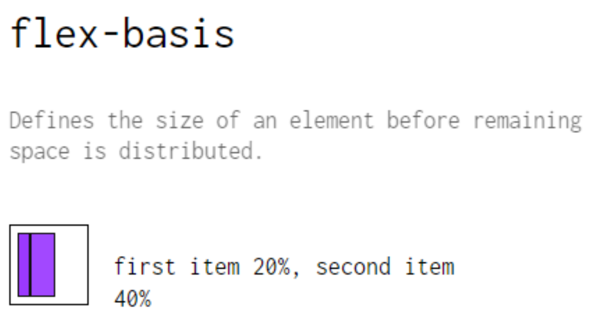
Allows you to determine how each child is allowed to grow as a part of a whole.


flex-grow: 1 (applied to all)


flex-grow (1, 2, and 3)

flex-basis

Defines the default size of an element before the remaining space is distributed. It can be a length (e.g., 20%, 5rem, etc.) or auto.



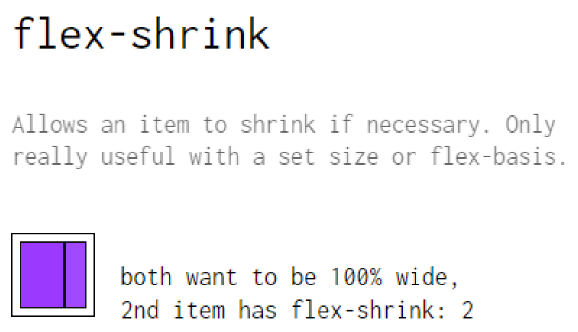
flex-basis

Defines the size of an element before remaining space is distributed.

first item 20%, second item 40%

flex-shrink

Defines the ability for a flex item to shrink if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.



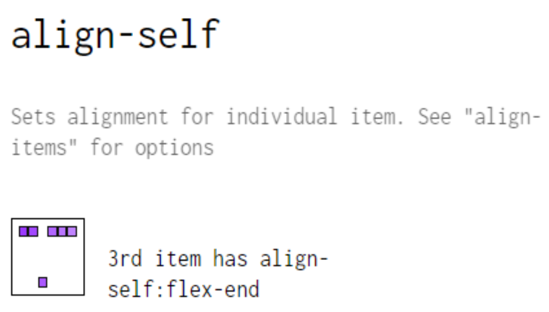
flex-shrink

Allows an item to shrink if necessary. Only really useful with a set size or flex-basis.

both want to be 100% wide, 2nd item has flex-shrink: 2

align-self

Allows the default alignment (or the one specified by align-items) to be overridden for individual flex items. Possible values: auto, flex-start, flex-end, center, baseline, stretch.



align-self

Sets alignment for individual item. See "align-items" for options

3rd item has align-self:flex-end

CSS Grid Layout

CSS Grid Layout is a powerful layout system designed for two-dimensional layouts. It allows for the precise placement of elements within rows and columns, making complex web designs easier to implement.

Key Concepts

Grid Container and Grid Items

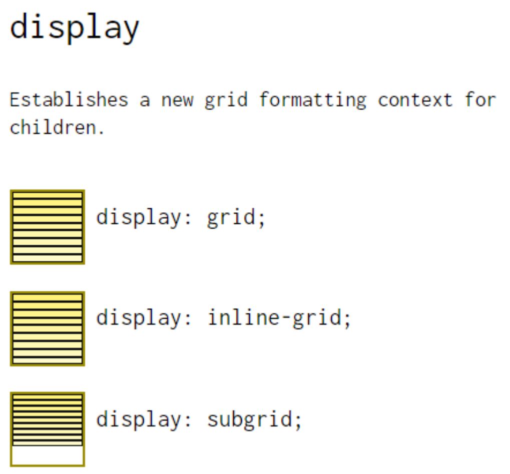
- Grid Container: The parent element that has display: grid or display: inline-grid.
- Grid Items: The direct children of a grid container.

Main Properties of CSS Grid

1. **Container Properties**

‘display’

- grid: Defines a block-level grid container.
- inline-grid: Defines an inline-level grid container.



display

Establishes a new grid formatting context for children.

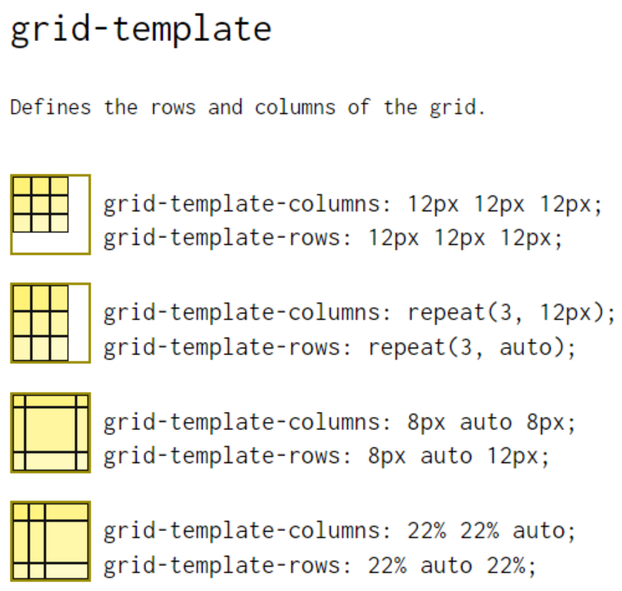
```
display: grid;
display: inline-grid;
display: subgrid;
```

grid-template-columns

- Defines the columns of the grid with a space-separated list of values. The values represent the track size (width of each column).
- Example: grid-template-columns: 100px 200px; creates two columns with widths of 100px and 200px.

grid-template-rows

- Defines the rows of the grid with a space-separated list of values. The values represent the track size (height of each row).
- Example: grid-template-rows: 100px 200px; creates two rows with heights of 100px and 200px.


grid-template

Defines the rows and columns of the grid.

```
grid-template-columns: 12px 12px 12px;
grid-template-rows: 12px 12px 12px;
```

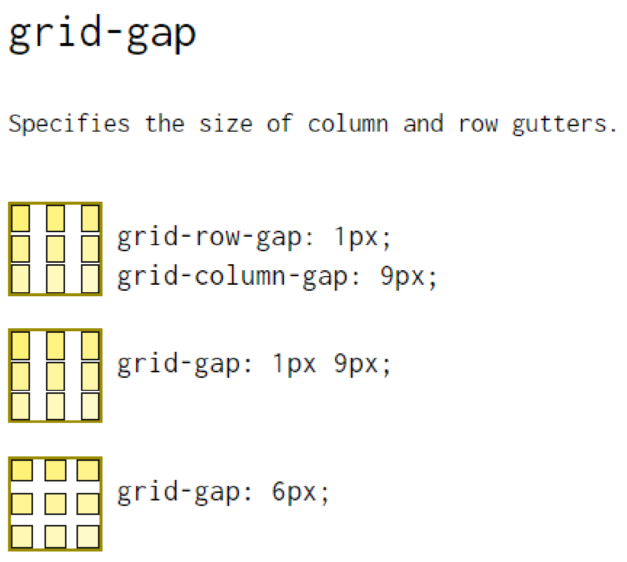
```
grid-template-columns: repeat(3, 12px);
grid-template-rows: repeat(3, auto);
```

```
grid-template-columns: 8px auto 8px;
grid-template-rows: 8px auto 12px;
```

```
grid-template-columns: 22% 22% auto;
grid-template-rows: 22% auto 22%;
```

grid-gap

- Shorthand for grid-row-gap and grid-column-gap.
- Example: grid-gap: 10px 20px; sets a 10px gap between rows and a 20px gap between columns.


grid-gap

Specifies the size of column and row gutters.

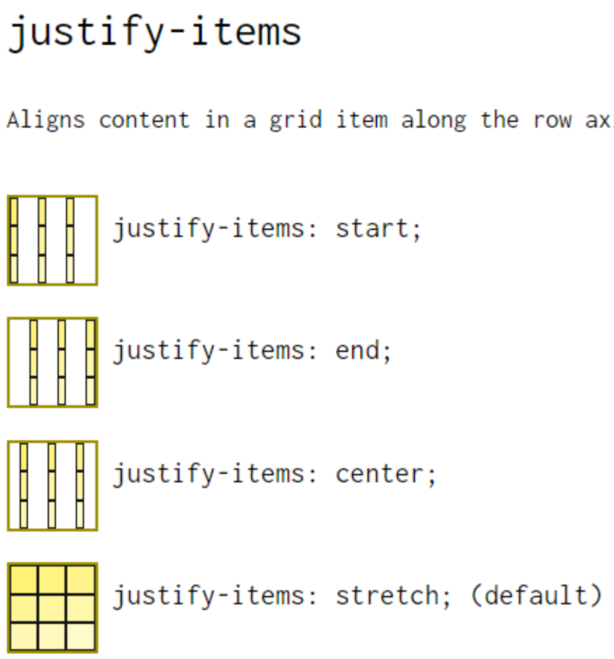
```
grid-row-gap: 1px;
grid-column-gap: 9px;
```

```
grid-gap: 1px 9px;
```

```
grid-gap: 6px;
```

justify-items

- Aligns grid items along the row axis (inline axis).
- Values: start, end, center, stretch.
- Example: justify-items: center; centers items horizontally within their grid area.



justify-items

Aligns content in a grid item along the row axis.

justify-items: start;

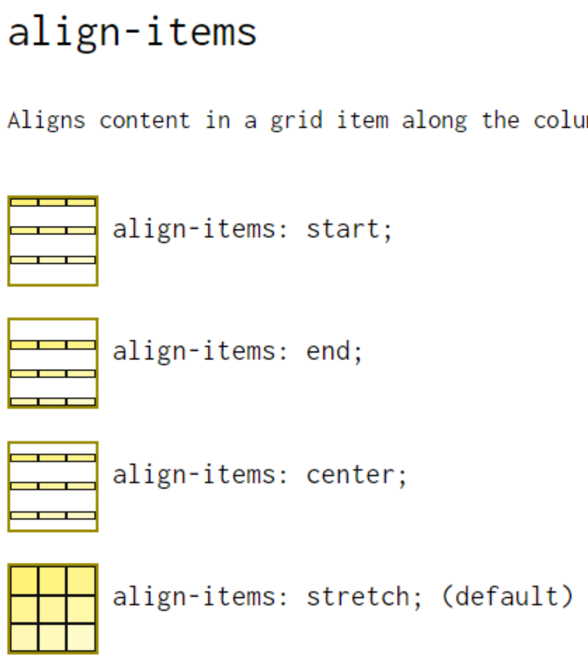
justify-items: end;

justify-items: center;

justify-items: stretch; (default)

align-items

- Aligns grid items along the column axis (block axis).
- Values: start, end, center, stretch.
- Example: align-items: center; centers items vertically within their grid area.



align-items

Aligns content in a grid item along the column axis.

align-items: start;

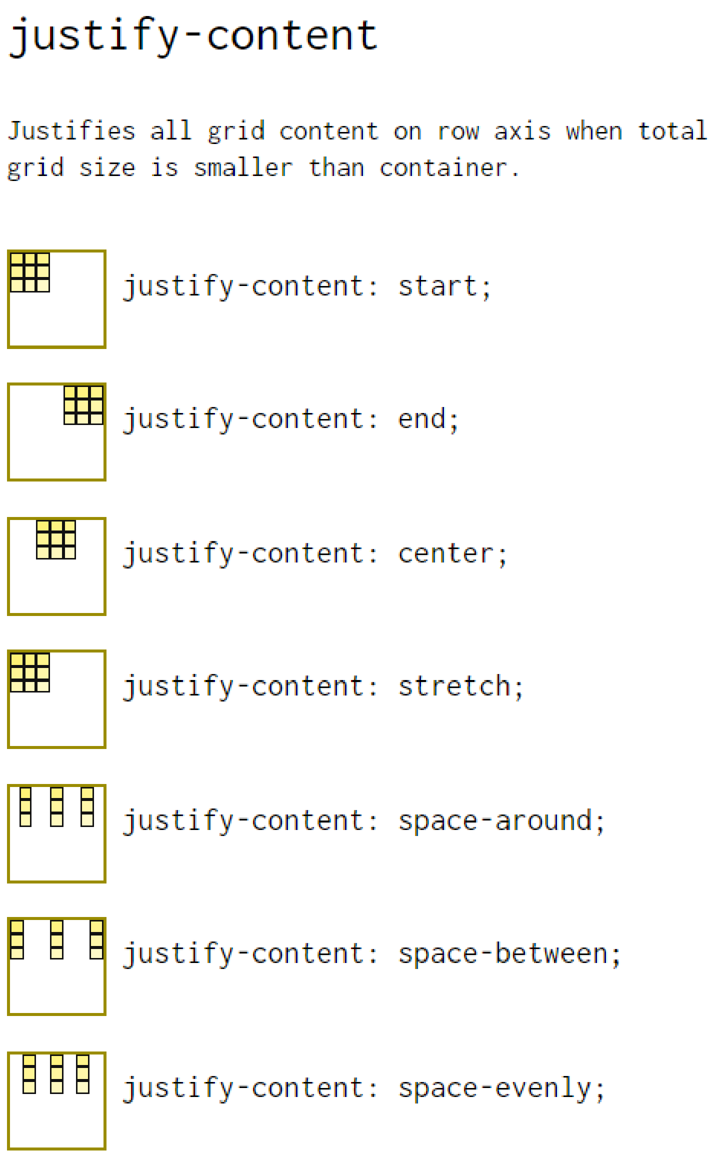
align-items: end;

align-items: center;

align-items: stretch; (default)

justify-content

- Aligns the grid container's content along the row axis.
- Values: start, end, center, space-between, space-around, space-evenly.
- Example: justify-content: space-between; distributes space between items.



justify-content

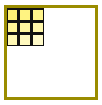
Justifies all grid content on row axis when total grid size is smaller than container.

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

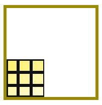
align-content

- Aligns the grid container's content along the column axis.
- Values: start, end, center, space-between, space-around, space-evenly, stretch.
- Example: align-content: space-around; distributes space around items.

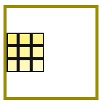
Justifies all grid content on column axis when total grid size is smaller than container.



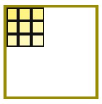
`align-content: start;`



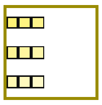
`align-content: end;`



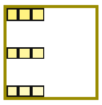
`align-content: center;`



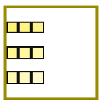
`align-content: stretch;`



`align-content: space-around;`



`align-content: space-between;`



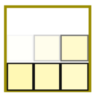
`align-content: space-evenly;`

grid-auto-flow Property

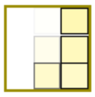
The grid-auto-flow property in CSS Grid Layout controls how auto-placed items are inserted into the grid. It specifies the algorithm that determines how these items are placed in the grid when there is no explicit position for them.

`grid-auto-flow`

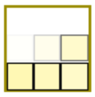
Algorithm for automatically placing grid items that aren't explicitly placed.



`grid-auto-flow: row;`



`grid-auto-flow: column;`

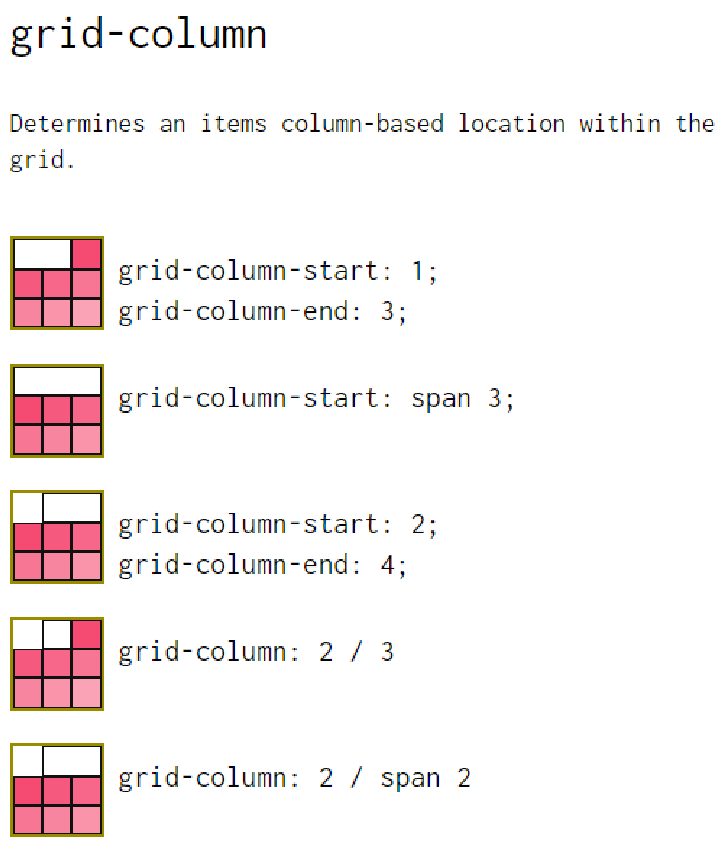


`grid-auto-flow: dense;`

2. Item Properties

grid-column

- Specifies the size and location of a grid item within the grid column structure.
- Example: grid-column: 1 / 3; makes the item span from the first to the third column.



grid-column

Determines an items column-based location within the grid.

```
grid-column-start: 1;
grid-column-end: 3;
```

```
grid-column-start: span 3;
```

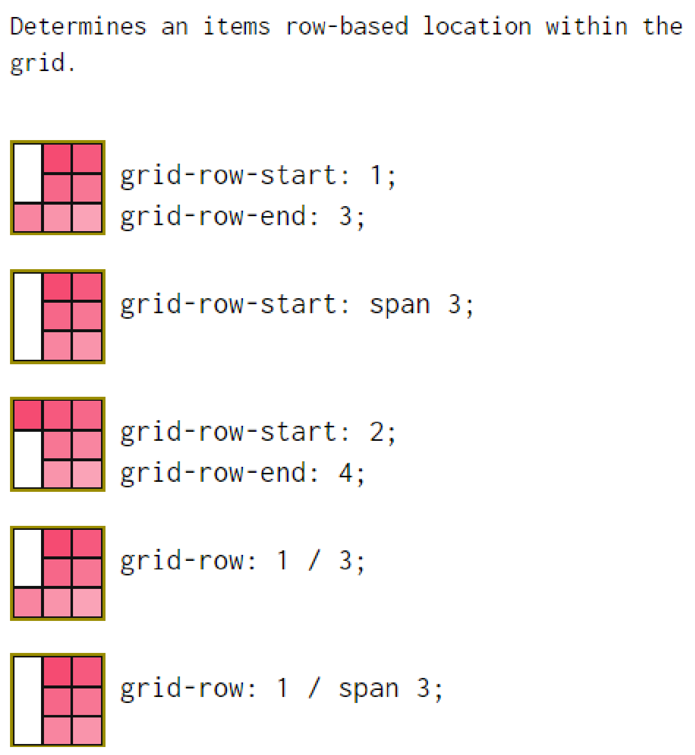
```
grid-column-start: 2;
grid-column-end: 4;
```

```
grid-column: 2 / 3
```

```
grid-column: 2 / span 2
```

grid-row

- Specifies the size and location of a grid item within the grid row structure.
- Example: grid-row: 1 / 2; makes the item span from the first to the second row.



Determines an items row-based location within the grid.

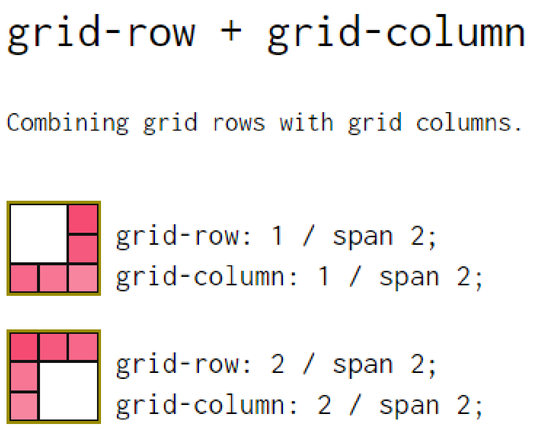
```
grid-row-start: 1;
grid-row-end: 3;
```

```
grid-row-start: span 3;
```

```
grid-row-start: 2;
grid-row-end: 4;
```

```
grid-row: 1 / 3;
```

```
grid-row: 1 / span 3;
```



grid-row + grid-column

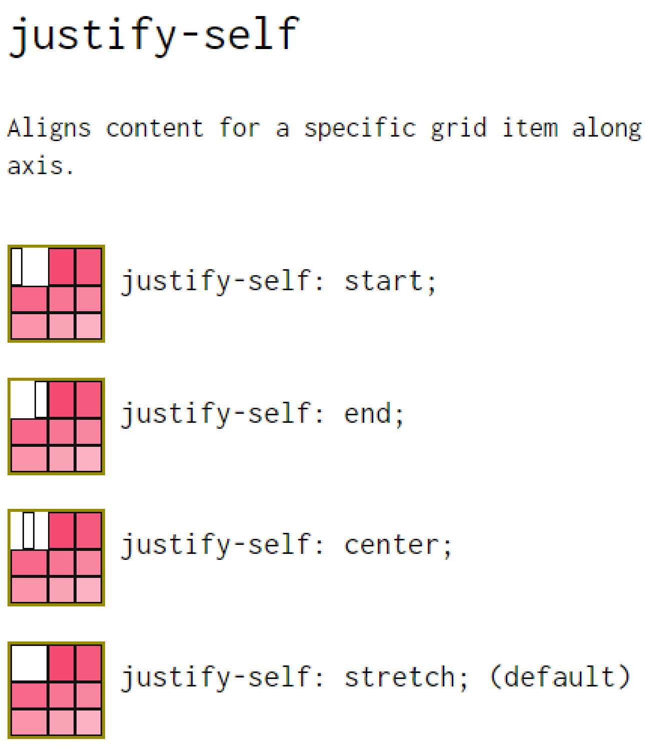
Combining grid rows with grid columns.

```
grid-row: 1 / span 2;
grid-column: 1 / span 2;
```

```
grid-row: 2 / span 2;
grid-column: 2 / span 2;
```

justify-self

- Aligns a grid item inside a cell along the row axis.
- Values: start, end, center, stretch.
- Example: justify-self: center; centers the item horizontally within its cell.



justify-self

Aligns content for a specific grid item along the row axis.

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

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

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

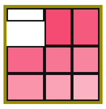
```
justify-self: stretch; (default)
```

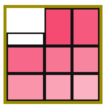
align-self

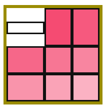
- Aligns a grid item inside a cell along the column axis.
- Values: start, end, center, stretch.
- Example: align-self: center; centers the item vertically within its cell.

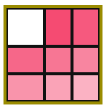
align-self

Aligns content for a specific grid item along the column axis.


align-self: start;


align-self: end;


align-self: center;


align-self: stretch; (default)