COESIONE ITALIA 21-27 PIEMONTE

Cofinanziato dall'Unione europea

REGIONE PIEMONTE

Web Developer

HTML, CSS e Strumenti di Digital Marketing (SEO, SEM, SEA)

Docente: Shadi Lahham

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

CSS Grid

Next-generation layout

Shadi Lahham - Web development

CSS Grid basics

What is CSS Grid

CSS Grid, a highly modern tool for building layouts in CSS, has been supported by all major browsers since 2017
Using the CSS grid layout HTML elements can be transformed into a grid container, enabling easy creation of responsive and structured designs

- Versatile grid system for web layouts
- Enables precise placement and alignment
- Supports intuitive reordering
- Facilitates responsive resizing
- Allows seamless content flow in multiple directions

Flexbox vs Grid

QUANDO CONVIENE USARE FLEX / GRID?

FLEX -> Il layout è monodimensionale (distribuire o allineare elementi lungo una riga o una colonna).

GRID -> Il layout è bidimensionale (distribuzione di elementi lungo righe e colonne).

- Flexbox
  - One-dimensional layout alignment along a single axis
  - Focuses on distributing space within a container
  - Great for responsive and dynamic layouts

- Grid
  - Two-dimensional layout control
  - Positions items anywhere within the grid
  - Enables complex and custom layouts

What's The Difference Between Flexbox And Grid?

CSS Grid vs. Flexbox: Which Should You Use and When?

Grid container & grid items

Grid container è un elemento che utilizza la proprietà CSS display: grid;. Questo stabilisce il contesto di grid per i suoi elementi figli.

a container defined using display: grid, organizing its direct children into a grid layout

Grid items I Grid Items sono gli elementi diretti del Grid Container. Ogni grid item è posizionato in celle della griglia, che vengono definite dal Grid Container

direct children of the grid container that become grid items, positioned within grid cells

Grid container & grid items

Basic grid

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

container padre con display: grid;

grid items che appartengono alla griglia con display grid

```
.grid-container {
  display: grid;
  grid-template-columns: 100px 100px 100px; /* 3 fixed-width columns */
  gap: 10px; /* gap between grid items */
}

.grid-item {
 border: 1px solid black; /* simple border for visual distinction */
}
```

Columns & rows

Grid template columns & rows

grid-template-columns
the number and size of the columns in the grid

grid-template-rows
the number and size of the rows in the grid

Syntax
uses units like px, %, em, or fr

rappresenta una frazione dello spazio disponibile all'interno di un grid container.

È una delle unità di misura più potenti in CSS Grid perché permette di distribuire lo spazio in modo flessibile

introduction to the fr unit
fr unit | mdn

Specifying rows and columns

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

```
.grid-container {
 display: grid;
 grid-template-columns: 100px 100px; /* 2 fixed-width columns */
 grid-template-rows: 50px 80px 35px; /* 3 fixed-height rows */
 gap: 10px;
}

.grid-item {
 border: 1px solid black;
}
```

Grid lines

aiutano a dividere uno spazio in righe e colonne, simili a quelle della carta millimetrata, per garantire precisione nelle misurazioni e nell'allineamento degli oggetti.

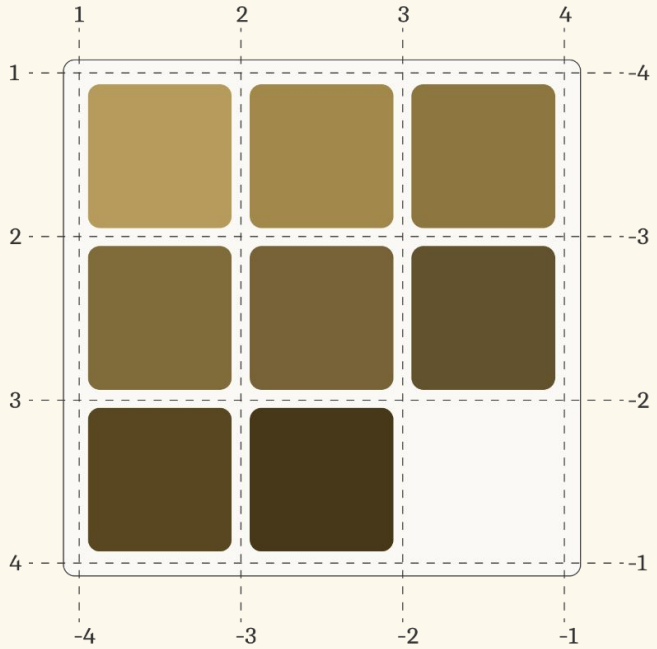
Grid lines

The lines separating the grid into rows and columns function like those on graph paper, aiding in accurate measurement and placement of items

Grid lines

Grid lines

Vertical column lines and horizontal row lines



Grid tracks, areas & cells

Grid tracks sono gli spazi tra le linee della griglia e rappresentano le righe orizzontali e le colonne verticali.
Essenzialmente, sono gli elementi stessi che costituiscono la griglia, formando la struttura di base.

The spaces between the grid lines are effectively the rows and columns themselves

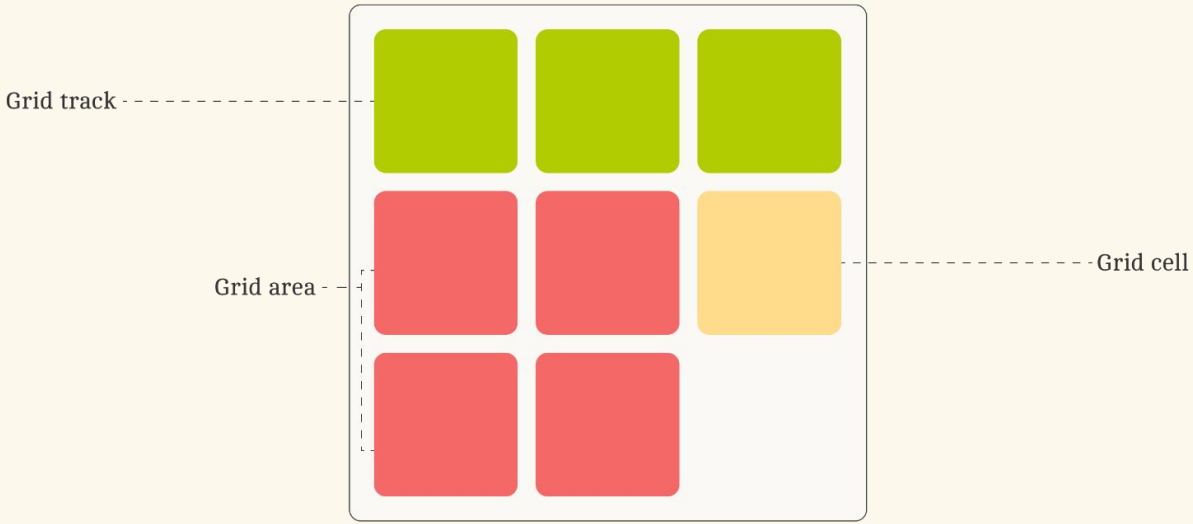
Grid areas sono spazi racchiusi da quattro linee della griglia
Queste aree formano regioni rettangolari che possono essere utilizzate per disporre e organizzare contenuti all'interno del layout.

The spaces enclosed by four grid lines, often spanning multiple grid cells, represent rectangular regions within the grid layout

Grid cells sono gli spazi rettangolari più piccoli creati dall'intersezione di righe e colonne.
Ogni cella può contenere contenuto (TESTO IMMAGINI ...)

The individual rectangular spaces created by the intersection of rows and columns, known as cells, can hold content

Grid lines, tracks & cells



Auto placement

è possibile sfruttare il posizionamento automatico, che permette di posizionare gli elementi senza specificare esattamente in quale cella devono andare.

Il browser gestisce il posizionamento degli elementi nella griglia

The grid can automatically place items without specifying the exact cell and by default, items will be placed in the next available spot

When a CSS Grid container has a specified height but no defined rows or columns, the grid items will automatically arrange themselves within the container, with the browser determining the number of rows based on the content and available space to ensure the items fit within the given height

The number of columns defaults to one unless specified otherwise, with each grid item adjusting its size to fit within the set height while maintaining the gaps defined by the gap property

Auto placement

```
/* When no rows and columns are specified but the grid has a set height,
  the grid items will auto-place within the container and adapt their sizes
  to fit within the specified height */

.grid-container {
 display: grid;
 gap: 10px;
 height: 350px; /* container has a fixed height - change to see behavior*/
}

.grid-item {
 border: 1px solid black;
}
```

Gaps

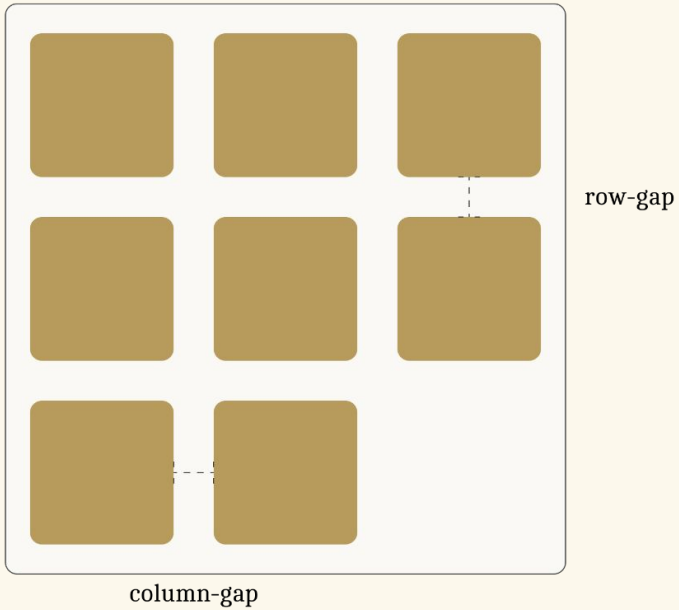
gap
sets the same spacing between rows and columns

row-gap
adjusts the vertical spacing between rows

column-gap
adjusts the horizontal spacing between columns

```
.grid-container {
  display: grid;
  grid-template-columns: 100px 100px 100px;
  row-gap: 30px; /* sets the vertical spacing between rows to 30px */
  column-gap: 10px; /* sets the horizontal spacing between columns to 10px */
}
```

Gaps



Grid sizing

Grid sizing

Fractional units (flexible value)
Allocates available space proportionally using **fr** units in CSS Grid

Min-Content si riferisce alla dimensione minima necessaria affinché un elemento possa essere visualizzato senza causare sovrapposizioni o tagli di contenuto.
Automatically sizes grid items based on the smallest possible content

Auto
Automatically sizes grid items based on their content or container size in CSS Grid

Repeat function
Creates multiple columns or rows with a repeated pattern in CSS Grid

Grid sizing

```
.grid-container {
  display: grid;

  grid-template-columns: repeat(3, 100px);
  /* 3 columns of 100px each */

  grid-template-columns: repeat(4, 1fr);
  /* 4 columns each taking up 1 fraction of available space */

  grid-template-rows: repeat(2, min-content);
  /* 2 rows sized based on the minimum content height */

  grid-template-columns: 1fr 2fr 1fr;
  /* 3 columns where the middle column is twice as wide */
}
```

Grid sizing

```
.grid-container {
  grid-template-columns: repeat(3, minmax(100px, 200px));
  /* 3 cols with min width 100px, max 200px */

  grid-template-rows: 100px repeat(2, 1fr);
  /* 3 rows: 1st row 100px, next 2 rows equal fractions */

  grid-template-columns: min-content 2fr 1fr;
  /* 3 cols 1st auto-sized, 2nd twice as wide as 3rd */

  grid-template-rows: repeat(2, auto);
  /* 2 rows each sized based on content */

  grid-template-columns: auto 1fr auto;
  /* 3 columns: 1st and 3rd auto-sized, middle column takes remaining space */
}
```

Placing items

Placing items

Items can be positioned in specific grid cells using the grid-column and grid-row shorthand properties, which define the starting and ending lines of the grid for precise placement

```
.item1 {
  grid-row: 1 / 3; /* starts at grid line 1, ends at grid line 3 */
  grid-column: 2 / 4; /* starts at grid line 2, ends at grid line 4 */
  /* spans across 2 rows and 2 columns */
}

.item2 {
  grid-row: 3 / 5; /* starts at grid line 3, ends at grid line 5 */
  grid-column: 2 / 5; /* starts at grid line 2, ends at grid line 5 */
  /* spans across 2 rows and 3 columns */
}
```

Placing items - single values

When using a single grid line value for grid-row and grid-column, the item starts at the specified line and ends at the next line, thereby occupying the row or column corresponding to the grid line

```
.item3 {
  grid-row: 2; /* starts at grid line 2, ends at grid line 3 */
  grid-column: 3; /* starts at grid line 3, ends at grid line 4 */
  /* occupies row 2 and column 3 */
}
```

Placing items - negative values

Negative values for grid-row and grid-column allow you to position items relative to the end of the grid, which is useful for placing items without knowing the exact number of grid lines

```
.item4 {
  grid-row: -3 / -1;
  /* starts at the third-to-last grid line, ends at the last grid line */

  grid-column: -4 / -2;
  /* starts at the fourth-to-last grid line, ends at the second-to-last grid line */

  /* spans across 2 rows and 2 columns from the end of the grid */
}
```

Placing items - precise control

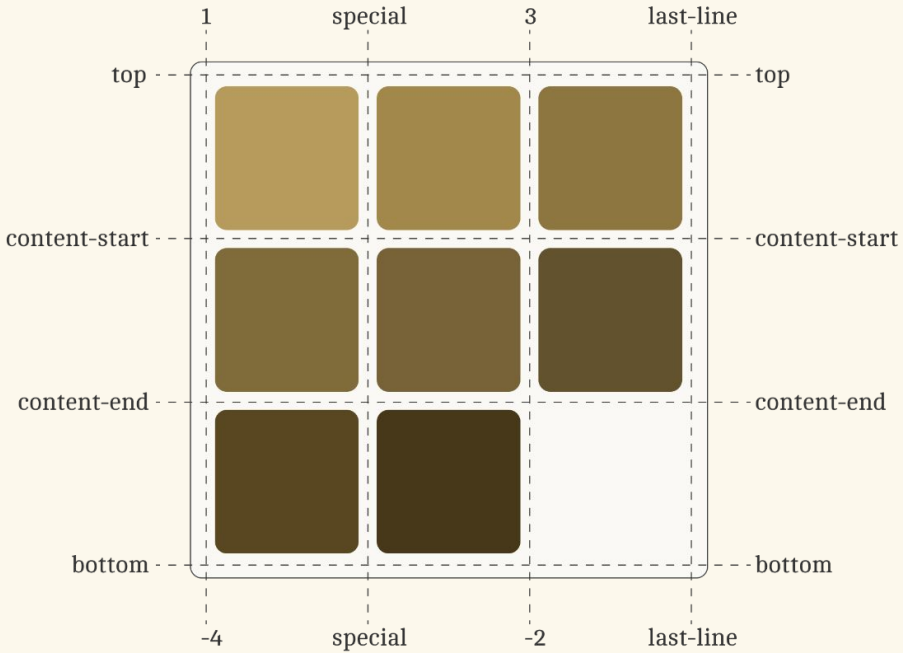
More precise control over grid item placement can be achieved by specifying the start and end points separately using grid-column-start, grid-column-end, grid-row-start, grid-row-end

```
.item3 {
  grid-column-start: 1; /* starts at grid line 1 */
  grid-column-end: 3; /* ends at grid line 3 */
  grid-row-start: 2; /* starts at grid line 2 */
  grid-row-end: 4; /* ends at grid line 4 */
  /* spans across columns 1 and 2, and rows 2 and 3 */
}

.item4 {
  grid-column-start: 3; /* starts at grid line 3 */
  grid-column-end: span 2; /* spans across 2 columns, ends at grid line 5 */
  grid-row-start: 1; /* starts at grid line 1 */
  grid-row-end: span 3; /* spans across 3 rows, ends at grid line 4 */
  /* spans across columns 3 to 5, and rows 1 to 3 */
}
```

Named grid lines

Naming grid lines

Grid lines of both columns and rows can be changed



Named grid lines

Named grid lines in CSS Grid allow for assigning names to specific grid lines, which enhances layout readability and simplifies positioning enabling intuitive placement of grid items using descriptive names rather than numerical values

CSS Grid syntax supports this by allowing names to be assigned within the grid-template-columns and grid-template-rows properties, which can then be used to position items within the grid

Named grid lines

```
<div class="container">
    <div class="item item1">item1</div>
    <div class="item item2">item2</div>
    <div class="item item3">item3</div>
</div>
```

Named grid lines

```
.container {
  display: grid;
  grid-template-columns: 1fr [special] 1fr 1fr [last-line];
  grid-template-rows: [top] 1fr [content-start] 1fr [content-end] 1fr [bottom];
  gap:10px;
}

.item {
  border: 1px solid black; padding:10px;
}
```

Named grid lines

```
.item1 {
  grid-column: 1 / special;
  grid-row: top / content-start;
}

.item2 {
  grid-column: special / last-line;
  grid-row: top / content-end;
}

.item3 {
  grid-column: 1 / last-line;
  grid-row: content-end / bottom;
}
```

Keyboard navigation & css grid

- CSS Grid affects presentation, not document structure
- Keyboard focus follows DOM order, not visual grid order
- Tabbing through grid elements may result in illogical focus movement

Possible fix

- Reorder grid elements in the DOM to match visual order
- This adjustment improves navigation consistency and user experience

Holy grail layout

The holy grail

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

The holy grail

```
.container {
  display: grid;
  grid-template-columns: 1fr 3fr; /* sidebar takes 1/4, main content 3/4 */
  grid-template-rows: auto 1fr auto; /* header and footer take as much space as needed */
  height: 100vh; /* full viewport height */
  gap: 10px; /* space between grid items */
}

.header {
  grid-column: 1 / 3; /* span across both columns */
  grid-row: 1; /* place in the first row */
  background-color: #fbda56;
}

.sidebar {
  grid-column: 1; /* place in the first column */
  grid-row: 2; /* place in the second row */
  background-color: #cf8e62;
}
```

The holy grail

```
.main {
  grid-column: 2; /* place in the second column */
  grid-row: 2; /* place in the second row */
  background-color: #dfe2d5;
  overflow-y:scroll;
}

.footer {
  grid-column: 1 / 3; /* span across both columns */
  grid-row: 3; /* place in the third row */
  background-color: #aacb77;
}
.header, .sidebar, .main, .footer {
  padding: 20px;
}
```

The holy grail - responsive

```
@media (max-width: 768px) { /* responsive design for smaller screens */
  .container {
    grid-template-columns: 1fr; /* single column layout */
    grid-template-rows: auto auto 1fr auto; /* stack all items */
  }

  .sidebar {
    grid-column: 1 / 3; /* sidebar takes full width */
    grid-row: 2; /* place below header */
  }

  .main {
    grid-column: 1 / 3; /* main content takes full width */
    grid-row: 3; /* place below sidebar */
  }
  .footer {
    grid-row: 4; /* place in the fourth row */
  }
}
```

Grid template areas

Grid template areas

The grid-template-areas property defines named grid areas that are easier to read and understand compared to using explicit grid-column and grid-row values

Each area, such as "header," "sidebar," "main," and "footer" correspond to sections in the HTML, making this method more intuitive by enabling a direct visualization of the layout from the CSS

The approach mirrors the structure of the HTML, making the arrangement of each section within the grid clear

The holy grail with template areas

```css
.container {
  display: grid;
  grid-template-areas:
    'header header' /* header spans across both columns */
    'sidebar main' /* sidebar on the left, main content on the right */
    'footer footer'; /* footer spans across both columns */
  grid-template-columns: 1fr 3fr; /* sidebar takes 1/4, main content 3/4 */
  grid-template-rows: auto 1fr auto; /* header and footer take as much space as needed */
  height: 100vh; /* full viewport height */
  gap: 10px; /* space between grid items */
}
```

The holy grail with template areas

```
.header {
  grid-area: header; /* assigns this element to the header area */
  background-color: #fbda56;
}

.sidebar {
  grid-area: sidebar; /* assigns this element to the sidebar area */
  background-color: #cf8e62;
}

.main {
  grid-area: main; /* assigns this element to the main content area */
  background-color: #dfe2d5;
  overflow-y: scroll;
}
```

The holy grail with template areas

```
.footer {
  grid-area: footer; /* assigns this element to the footer area */
  background-color: #aacb77;
}

.header, .sidebar, .main, .footer {
  padding: 20px;
}
```

Responsive Adjustments

In media queries, grid-template-areas are redefined to stack elements vertically on smaller screens, simplifying layout adjustments without the need to change individual grid positions manually

Using grid-template-areas simplifies the layout management and improves code readability, especially when dealing with complex layouts.

The holy grail with template areas

```
/* responsive design for smaller screens */
@media (max-width: 768px) {
  .container {
    grid-template-areas:
      'header' /* header on top */
      'sidebar' /* sidebar below header */
      'main' /* main content below sidebar */
      'footer'; /* footer at the bottom */
    grid-template-columns: 1fr; /* single column layout */
    grid-template-rows: auto auto 1fr auto; /* stack all items */
  }
}
```

Grid alignment

align-items

align-items in CSS Grid is used to align items along the vertical axis within their grid area

It controls how content is placed within the grid cells, determining whether items should be aligned at the start, center, end, or stretched to fill the grid cell's height

Values

- start: aligns items to the start of the grid area
- end: aligns items to the end of the grid area
- center: centers items within the grid area
- stretch - default: stretches items to fill the grid area

justify-items

justify-items in CSS Grid is used to align items along the horizontal axis within their grid area

It determines whether items are aligned to the start, center, end, or stretched to fill the grid cell's width

Values

- start: aligns items to the start of the grid area
- end: aligns items to the end of the grid area
- center: centers items within the grid area
- stretch - default: stretches items to fill the grid area

place-items

place-items in CSS Grid is a shorthand property combining **align-items** and **justify-items**, setting both vertical and horizontal alignment of grid items within their grid areas

If only one value is provided, it applies to both axes, but the value must be valid for both, or the whole property is invalid

Values

- start: aligns items to the start of both axes
- end: aligns items to the end of both axes
- center: centers items on both axes
- stretch: stretches items to fill their grid area on both axes

Item alignment example

```
<div class="container">
    <!-- fake contents to make grid areas visible for this example -->
    <div class="fake pos-1"></div>
    <div class="fake pos-2"></div>
    <div class="fake pos-3"></div>
    <div class="fake pos-4"></div>
    <div class="fake pos-5"></div>
    <div class="fake pos-6"></div>
    <!-- actual content -->
    <div class="item pos-1">item 1</div>
    <div class="item pos-2">item 2</div>
    <div class="item pos-3">item 3</div>
    <div class="item pos-4">item 4</div>
    <div class="item pos-5">item 5</div>
    <div class="item pos-6">item 6</div>
</div>
```

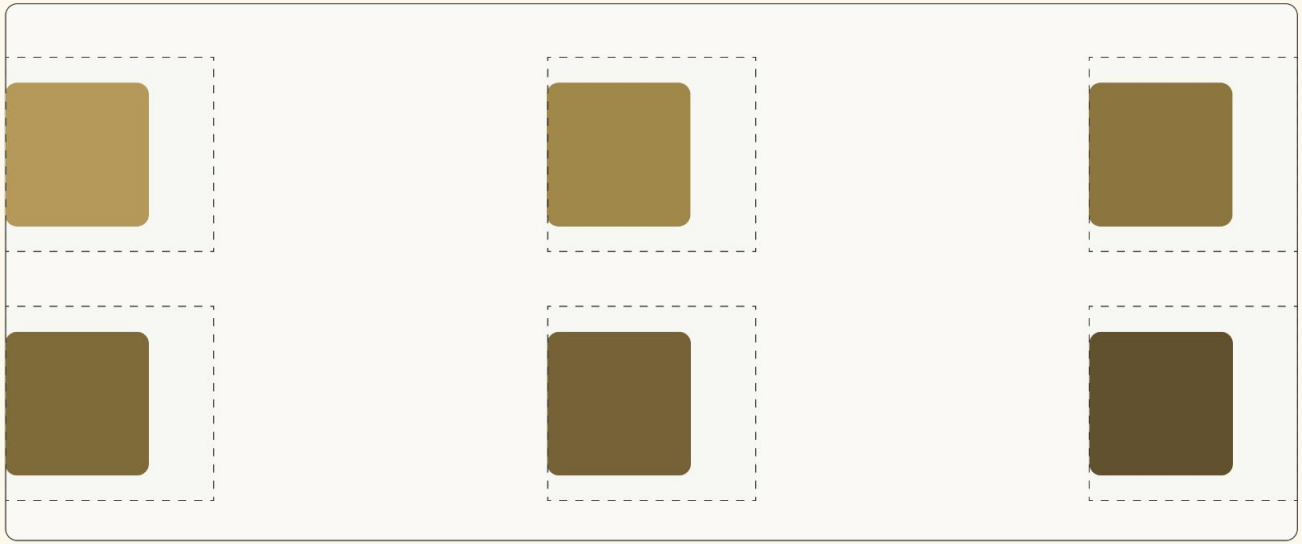
Item alignment example

```
.container {
  display: grid;
  grid-template: repeat(2, 90px) / repeat(3, 100px); /* shorthand rows / columns */
  justify-items: start; /* change to start, center, end, or stretch to see effect */
  align-items: center; /* change to start, center, end, or stretch to see effect */
  justify-content: space-between;
  align-content: space-evenly;
  height: 250px;
  background-color: lightgreen;
}

.item {
  background-color: lightblue;
}

.fake {
  place-self: stretch;
  background-color: lightsalmon;
}
```

Item alignment example

```
/* forced positioning to overlap fake and real contents */
.pos-1 { grid-area: 1/1; }
.pos-2 { grid-area: 1/2; }
.pos-3 { grid-area: 1/3; }
.pos-4 { grid-area: 2/1; }
.pos-5 { grid-area: 2/2; }
.pos-6 { grid-area: 2/3; }
```

Item alignment example



align-content

align-content in CSS Grid is used to align the entire grid along the vertical within the container

This property applies only when the grid's total height is less than the container height and controls the spacing between rows

Values

- start: aligns the grid rows to the start of the container
- end: aligns the grid rows to the end of the container
- center: centers the grid rows within the container
- stretch: stretches the grid rows to fill the container
- space-between: distributes grid rows evenly with the first grid rows at the start and the last grid rows at the end
- space-around: distributes grid rows evenly with equal space around them
- space-evenly: distributes grid rows so that the space between any two rows and the space to the edges is equal

justify-content

justify-content in CSS Grid is used to align the entire grid along the horizontal axis within the container

It determines how the space between and around grid items is distributed when the grid container is larger than the grid itself

Values

- start: aligns the grid columns to the start of the container
- end: aligns the grid columns to the end of the container
- center: centers the grid columns within the container
- stretch: stretches the grid columns to fill the container
- space-between: distributes grid columns evenly with the first grid columns at the start and the last grid columns at the end
- space-around: distributes grid columns evenly with equal space around them
- space-evenly: distributes grid columns so that the space between any two grid columns and the space to the edges is equal

place-content

place-content is a shorthand for **align-content** and **justify-content**, controlling the alignment and spacing of the entire grid along both vertical and horizontal axes

If only one value is specified, it applies to both, but it must be valid for both axes, or the whole property is invalid

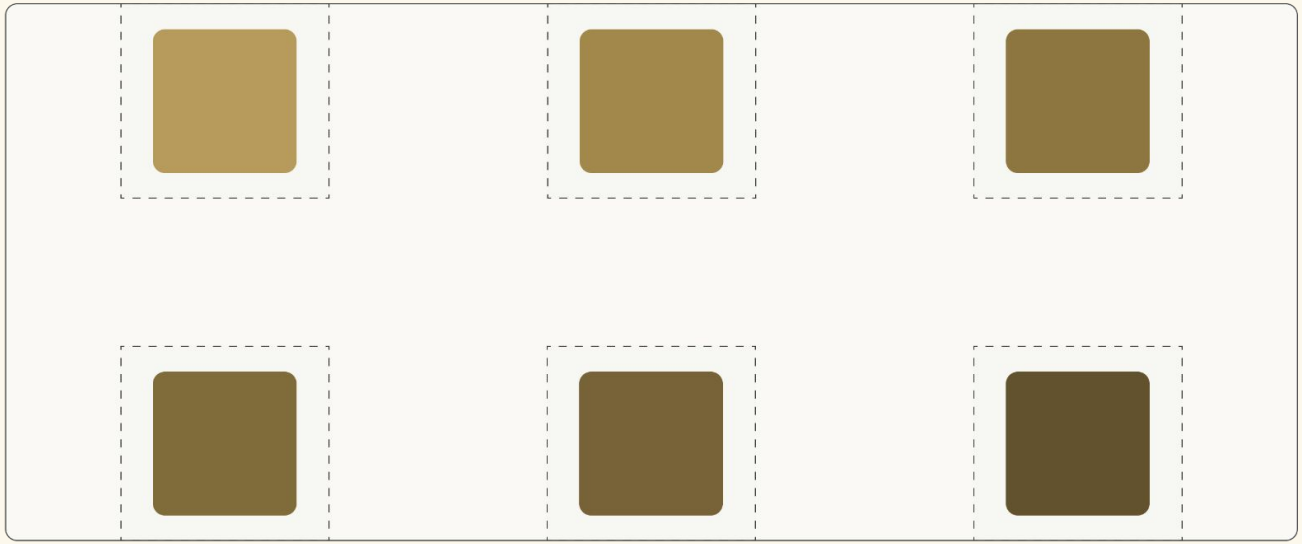
Values

- start: aligns the grid to the start of both axes
- end: aligns the grid to the end of both axes
- center: centers the grid on both axes
- stretch: stretches the grid to fill the container on both axes
- space-between: distributes grid rows and columns evenly with the first and last grid rows and columns at the edges
- space-around: distributes grid rows and columns with equal space around them
- space-evenly: distributes grid rows and columns with equal space between them and to the edges

Content alignment example

```css
/* make the following changes to the previous example */
.container {
  display: grid;
  grid-template: repeat(2, 90px) / repeat(3, 100px); /* shorthand rows / columns */
  justify-items: center;
  align-items: center;
  justify-content: space-around; /* change to see the effect */
  align-content: space-between; /* change to see the effect */
  height: 250px;
  background-color: lightgreen;
}
```

Content alignment example



Start & stretch difference

```css
/* changes to the previous example to illustrate stretch */
.container {
  display: grid;
  grid-template: repeat(2, 90px) / repeat(2, 100px) auto; /* last column is auto */
  justify-items: center;
  align-items: center;
  justify-content: stretch; /* change stretch to start to see the effect */
  gap: 10px;
  background-color: lightgreen;
}
```

Individual grid item alignment

The CSS properties place-self, align-self, and justify-self are used to control the alignment of individual grid items within a grid container

These properties work similarly to align-items, justify-items, and place-items, but they apply to individual grid items rather than all items in the grid

align-self
aligns a single grid item along the vertical axis

justify-self
aligns a single grid item along the horizontal axis

place-self
a shorthand for setting both **align-self** and **justify-self**

Individual grid item alignment

```
<div class="grid-container">
    <div class="grid-item">item 1</div>
    <div class="grid-item align-self-example">align self</div>
    <div class="grid-item justify-self-example">justify self</div>
    <div class="grid-item place-self-example">place self</div>
    <div class="grid-item">item 5</div>
</div>
```

Individual grid item alignment

```
.grid-container {
  display: grid;
  grid-template-columns: 100px 100px 100px;
  grid-template-rows: 100px 100px;
  gap: 10px;
  height: 250px;
  border: 1px solid black;

  /* these properties apply to all items by default */
  align-items: start; /* aligns items at the start (top) of their grid area */
  justify-items: start; /* aligns items at the start (left) of their grid area */
}

.grid-item {
  background-color: lightblue;
  text-align: center;
  border: 1px solid gray;
}
```

Individual grid item alignment

```css
/* overrides align-items for a single item, aligning it at the bottom */
.align-self-example {
  align-self: end;
}

/* overrides justify-items for a single item, centering it horizontally */
.justify-self-example {
  justify-self: center;
}

/* overrides both align-items and justify-items for a single item */
.place-self-example {
  place-self: center end; /* centers vertically, aligns right horizontally */
}
```

Auto rows and columns

grid-auto-rows

grid-auto-rows in CSS Grid is used to define the size of implicitly created grid rows

When items are placed in a grid area that doesn't have a predefined row, the browser generates a new row, and grid-auto-rows controls its height

Values

- <length>: fixed height (e.g., 100px)
- <percentage>: height relative to the grid container (e.g., 20%)
- auto: height based on content
- min-content: minimum size to fit the content
- max-content: maximum size to fit the content
- minmax(min, max): size range with a minimum and maximum value
- <flex>: specifies a flexible size that grows or shrinks relative to the available space

grid-auto-columns

grid-auto-columns in CSS Grid sets the size of implicitly created columns

When the grid container has more items than the defined columns, new columns are automatically created based on this property

Values

- <length>: fixed width (e.g., 100px)
- <percentage>: width relative to the grid container (e.g., 20%)
- auto: width based on content
- min-content: minimum size to fit the content
- max-content: maximum size to fit the content
- minmax(min, max): size range with a minimum and maximum value
- <flex>: specifies a flexible width that grows or shrinks relative to the available space

grid-auto-columns example

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

grid-auto-columns example

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(2, 120px); /* define two columns explicitly */
  grid-auto-columns: 40px; /* auto-generated columns will have a width of 40px */
  grid-gap: 10px;

.grid-item {
  background-color: lightblue;
}

.grid-item:nth-child(4) {
  grid-column: 4;
  /* placement creates additional implicit columns */
}

/* note: it's not obligatory to define all rows and columns */
```

Auto flow

grid-auto-flow

grid-auto-flow in CSS Grid controls how the grid auto-places items into the grid when explicit grid lines are not defined

It determines the direction and order in which items are placed in the grid

Values

- row: items are placed by filling rows first before moving to the next row
- column: items are placed by filling columns first before moving to the next column
- dense: items are placed in a dense packing order, filling in gaps more efficiently
- row dense: items are placed in rows with dense packing
- column dense: items are placed in columns with dense packing

grid-auto-flow example

```
.grid-container {
  display: grid;
  grid-template-columns: 80px 80px;
  gap: 10px;
  grid-auto-flow: column; /* try change to row */
}

.grid-item {
  border: 1px solid black;
  padding: 10px;
}
```

Grid shorthand

grid shorthand

grid is a shorthand property in CSS Grid that provides a concise way to define the grid layout by setting the values for the grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, and grid-auto-flow properties

Syntax

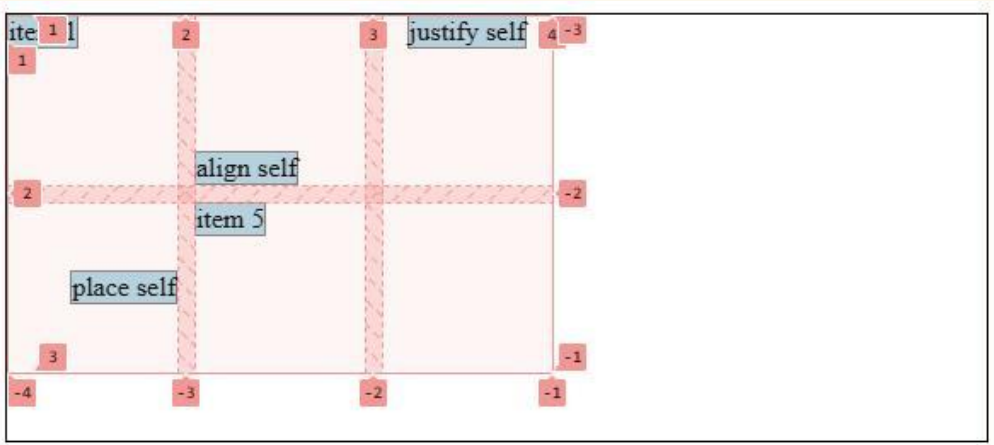
- grid: [<grid-template-rows> / <grid-template-columns>] [<grid-template-areas>] [<grid-auto-rows>] [<grid-auto-columns>] [<grid-auto-flow>]

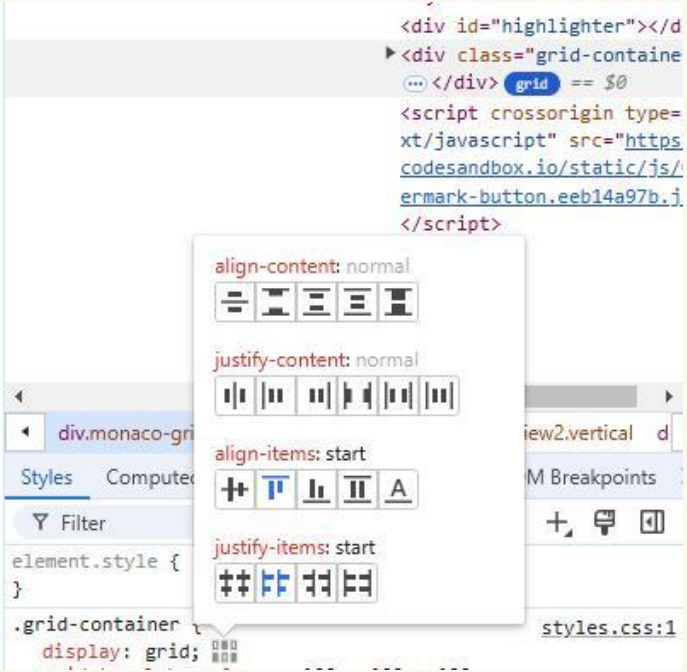
Examples

```
.grid-container {
  display: grid;
  grid: 100px 200px / auto auto; /* sets fixed row heights and column widths with auto placement */
}

.grid-container {
  display: grid;
  grid: "header header" "sidebar content" / 1fr 3fr; /* defines a grid with areas and columns */
}
```

Debug tools

Chrome grid debug tools





Your turn

1.Grid garden

Complete all levels of the following game for learning CSS grid

Grid Garden

Submit an .md file with the answer for each level

2.Grid art

Create a dynamic image gallery using CSS Grid which showcases images with random sizes and placements that change over time, with interactive features

Layout

- use css grid to create a responsive gallery
- images should be displayed in various sizes and randomly placed within the grid

Behavior

- the layout of the images should change every n seconds
- on mouseover, the image grid should freeze and stop changing
- allow users to click on images to enlarge them

Bonus

- implement smooth animations or transitions to the gallery - see the following references
  - animating CSS grid examples
  - animate-css-grid
- provide navigation for browsing through the images
  - e.g. next/previous buttons or keyboard navigation

References

Excellent guide

An Interactive Guide to CSS Grid

CSS tricks guides

CSS Grid Layout Guide

MDN guide

Basic concepts of grid layout

References

Basic grid concepts

grid

grid-template-columns

grid-template-rows

gap

References

Sizing

min-content

the fr unit

repeat()

minmax()

References

Placement

grid-column

grid-row

grid-column-start

grid-column-end

grid-row-start

grid-row-end

Layout using named grid lines

grid-template-areas

References

Tools

Interactive Grid generator

The holy grail layout

The Holy Grail Layout With CSS Grid

References

Additional resources

Usage examples of CSS Grid Layout

CSS Grid vs. Flexbox: Which Should You Use and When?

The Difference Between Explicit And Implicit Grids

Masonry layout - MDN - experimental

Video lessons

Free Learn Grid Layout video series

CSS Grid