FWDP 1000 – Day 8

Course:	Web Development 1
Instructor:	Gabbie Bade

Morning Review

- Download the files.
- Open the CSS file in your code editor.
- Complete the four tasks described in the comments of the CSS file.
- Set up your day-8 branch.

We will go over this in 10 minutes.

Text over Images

Read this two-part article about designing accessible text over images.

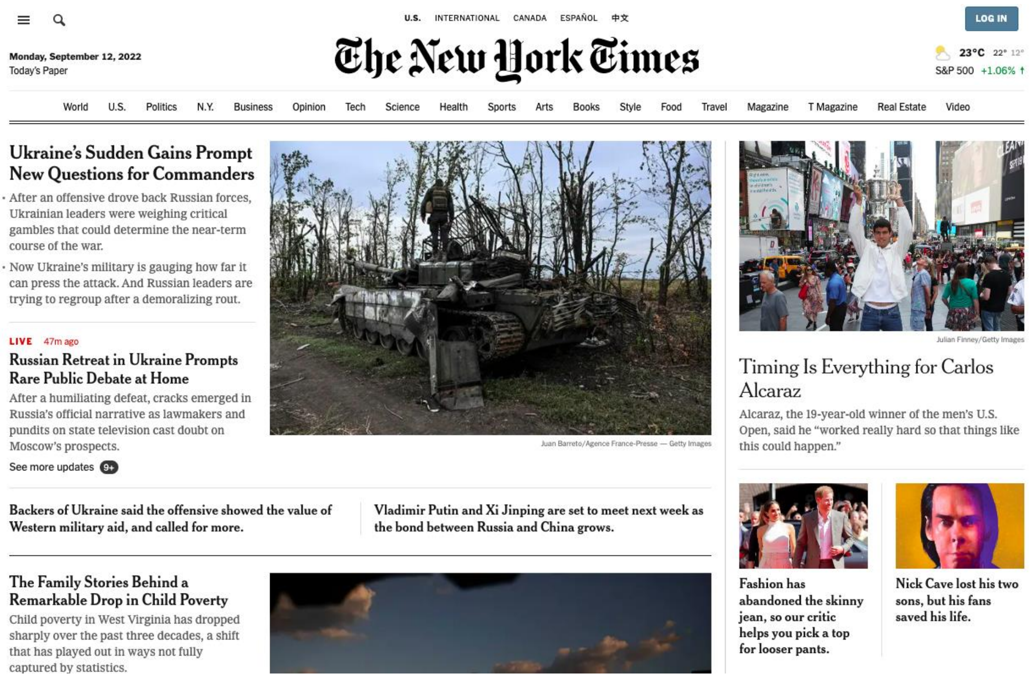
It has tons of great examples worth looking at for inspiration.

https://www.smashingmagazine.com/2023/08/designing-accessible-text-over-images-part1/

https://www.smashingmagazine.com/2023/08/designing-accessible-text-over-images-part2/

Agenda

- CSS Grid
- Assignment #6



U.S. INTERNATIONAL CANADA ESPAÑOL 中文 LOG IN

Monday, September 12, 2022
Today's Paper

The New York Times

23°C 22° 12°
S&P 500 +1.06% ↑

World U.S. Politics N.Y. Business Opinion Tech Science Health Sports Arts Books Style Food Travel Magazine T Magazine Real Estate Video

Ukraine's Sudden Gains Prompt New Questions for Commanders

- After an offensive drove back Russian forces, Ukrainian leaders were weighing critical gambles that could determine the near-term course of the war.
- Now Ukraine's military is gauging how far it can press the attack. And Russian leaders are trying to regroup after a demoralizing rout.

Juan Barreto/Agence France-Presse — Getty Images

LIVE 47m ago

Russian Retreat in Ukraine Prompts Rare Public Debate at Home

After a humiliating defeat, cracks emerged in Russia's official narrative as lawmakers and pundits on state television cast doubt on Moscow's prospects.

See more updates 9+

Backers of Ukraine said the offensive showed the value of Western military aid, and called for more.

Vladimir Putin and Xi Jinping are set to meet next week as the bond between Russia and China grows.

The Family Stories Behind a Remarkable Drop in Child Poverty

Child poverty in West Virginia has dropped sharply over the past three decades, a shift that has played out in ways not fully captured by statistics.

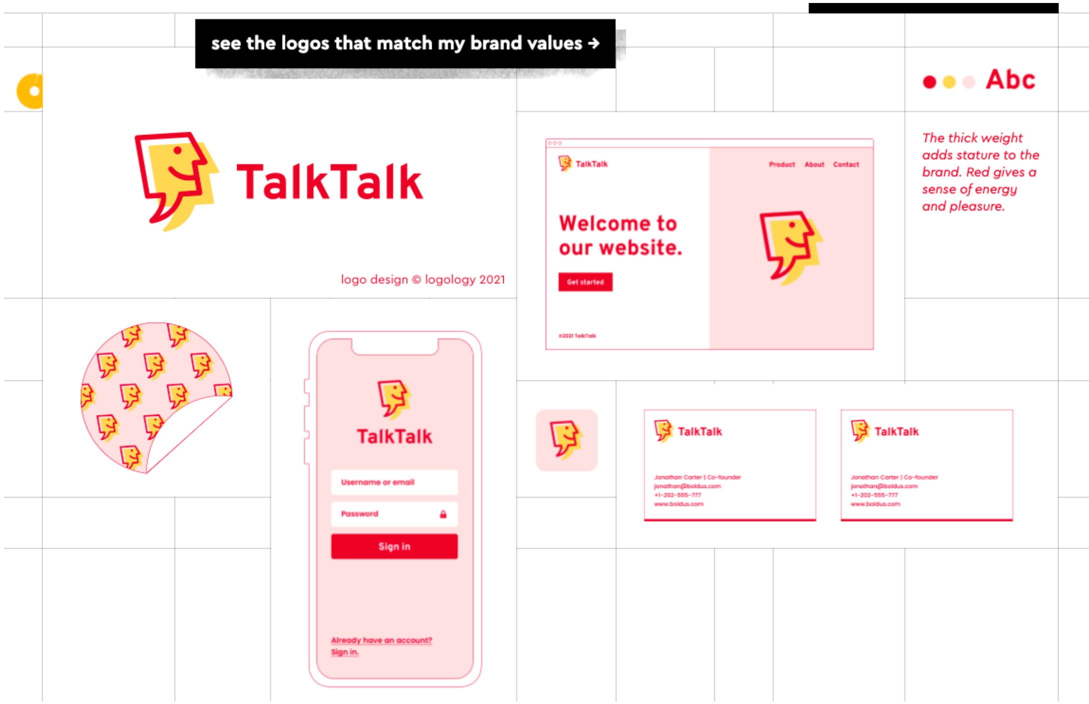
Julian Finney/Getty Images

Timing Is Everything for Carlos Alcaraz

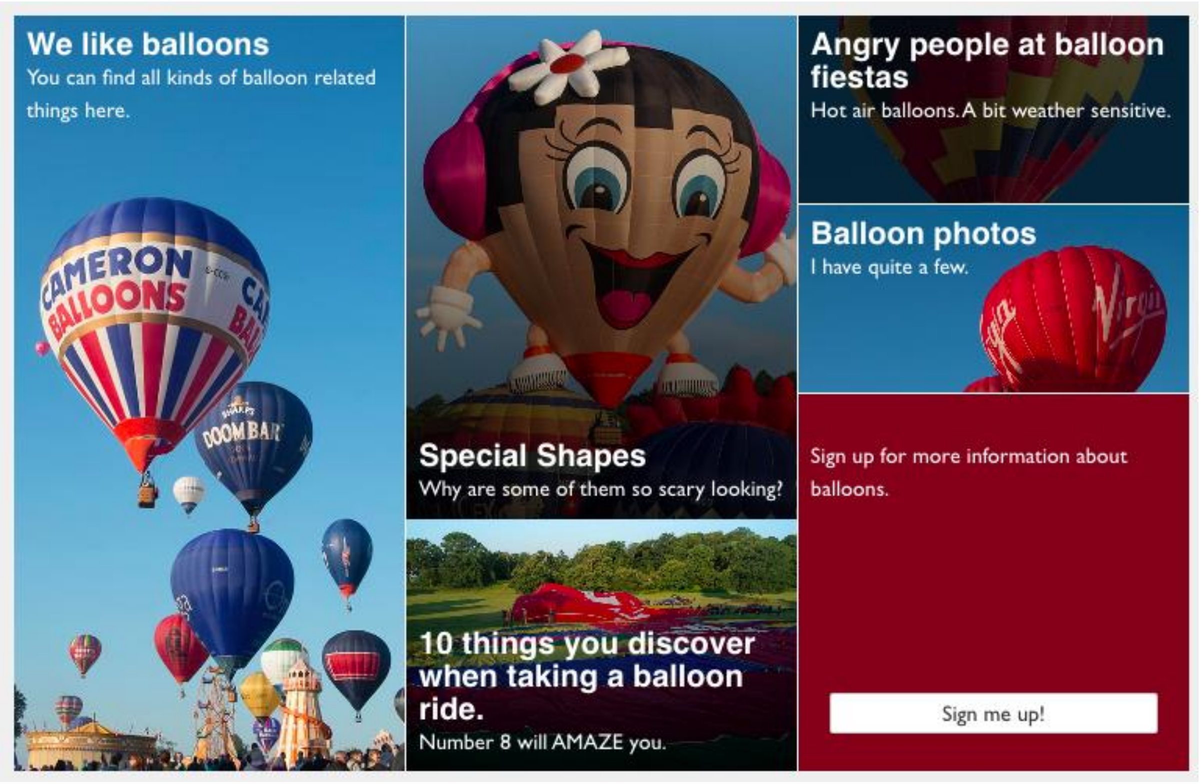
Alcaraz, the 19-year-old winner of the men's U.S. Open, said he "worked really hard so that things like this could happen."

Fashion has abandoned the skinny jean, so our critic helps you pick a top for looser pants.

Nick Cave lost his two sons, but his fans saved his life.



see the logos that match my brand values →
Abc
TalkTalk
logo design © logology 2021
The thick weight adds stature to the brand. Red gives a sense of energy and pleasure.
TalkTalk
Product About Contact
Welcome to our website.
Get started
©2021 TalkTalk
TalkTalk
Username or email
Password
Sign in
Already have an account? Sign in.
TalkTalk
Jonathan Carter | Co-founder
jonathan@boldus.com
+1-202-555-777
www.boldus.com
TalkTalk
Jonathan Carter | Co-founder
jonathan@boldus.com
+1-202-555-777
www.boldus.com



We like balloons
You can find all kinds of balloon related things here.

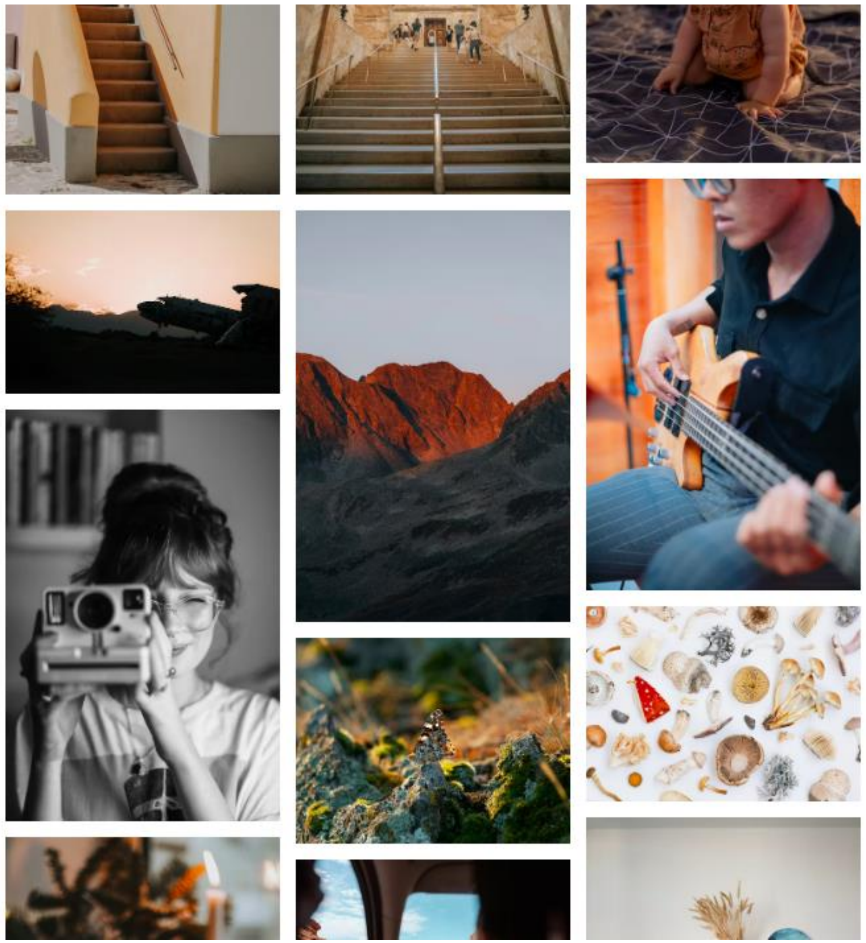
Special Shapes
Why are some of them so scary looking?

10 things you discover when taking a balloon ride.
Number 8 will AMAZE you.

Angry people at balloon fiestas
Hot air balloons. A bit weather sensitive.

Balloon photos
I have quite a few.

Sign up for more information about balloons.



CSS Grid Examples

https://www.nytimes.com/

https://unsplash.com/

https://www.logology.co/

https://codepen.io/rachelandrew/pen/QKwvxJ

https://codepen.io/adrianroworth/pen/OpeyZq

CSS Grid

CSS Grid

CSS Grid Layout, or “Grid”, is a two-dimensional layout system that lays out content in rows and columns.

In contrast, Flexbox is meant for one-dimensional layouts: columns **or** rows.

Thankfully, they use some common CSS properties for justifying and aligning items.

Grid vs Flex

Flexbox with `flex-wrap` **can** create multiple columns / rows. But if you find yourself trying to align items in those second columns or rows… switch to using Grid.

Flexbox is best for controlling a single column or row.

Grid is best for controlling multiple columns and rows.

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Relationship_of_Grid_Layout

Browser Support

Flexbox is older and better supported by browsers than Grid.

The difference is small and growing smaller every year:

https://caniuse.com/css-grid

https://caniuse.com/flexbox

@supports

If you want to make sure the browser supports a CSS feature, you can use the `@supports` feature query in CSS.

```
@supports (display: grid) {
    div {
        display: grid;
    }
}
```

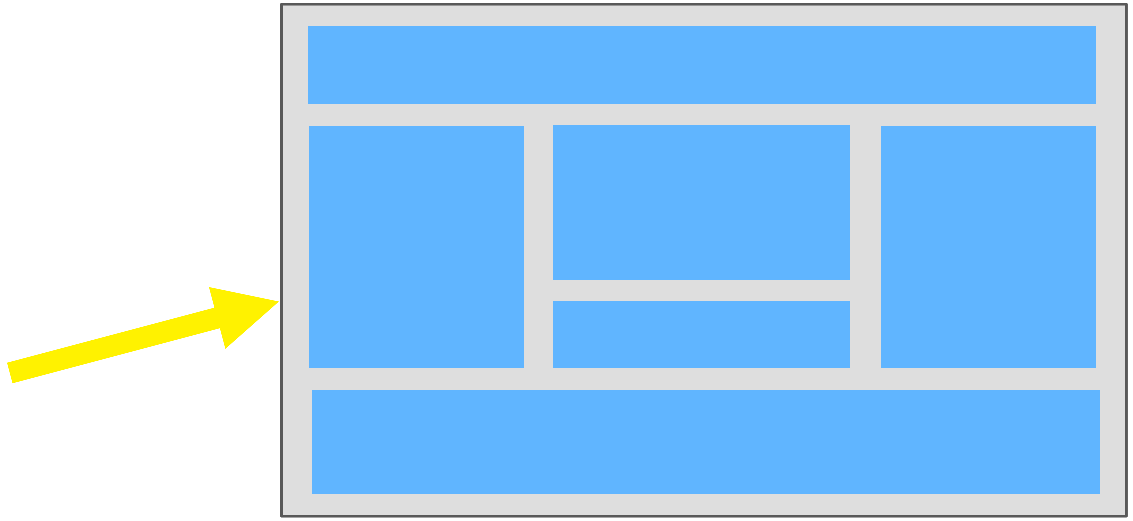
https://developer.mozilla.org/en-US/docs/Web/CSS/@supports

Grid Terminology

- Grid Container
- Grid Item
- Grid Line
- Grid Track
- Grid Cell
- Grid Area

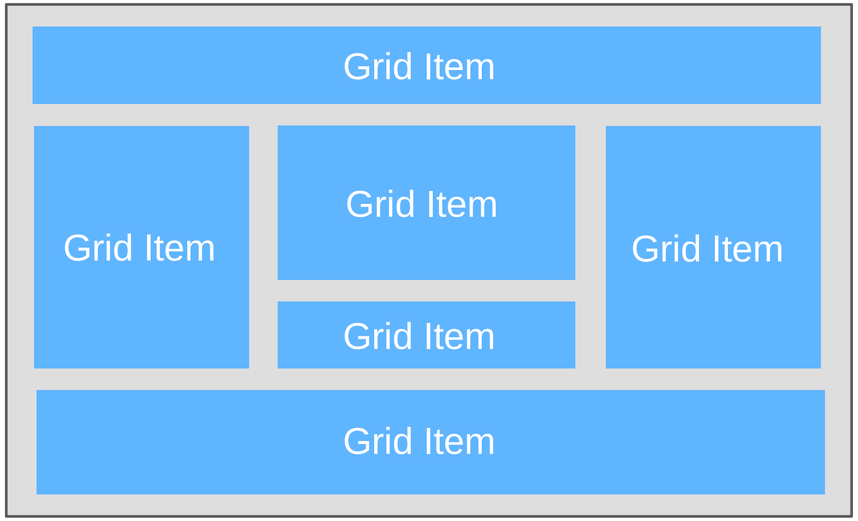
Grid Container

The **Grid Container** is the element that surrounds the grid items. In CSS, it is on the element you set `display: grid`.



Grid Item

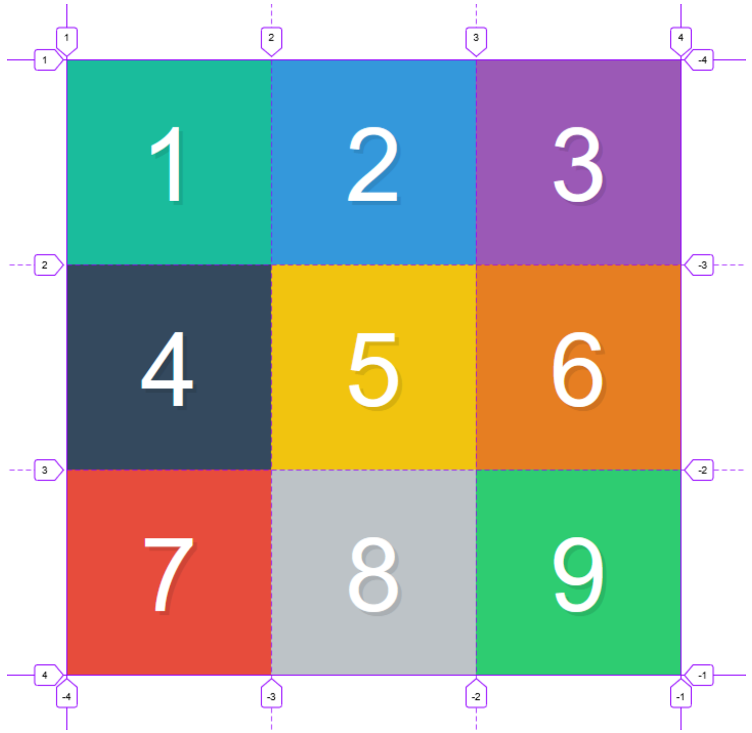
The **Grid Items** are all of the direct children of a grid container.



Grid Line

The **Grid Lines** are the lines that make up the structure of the Grid and they are both horizontal and vertical.

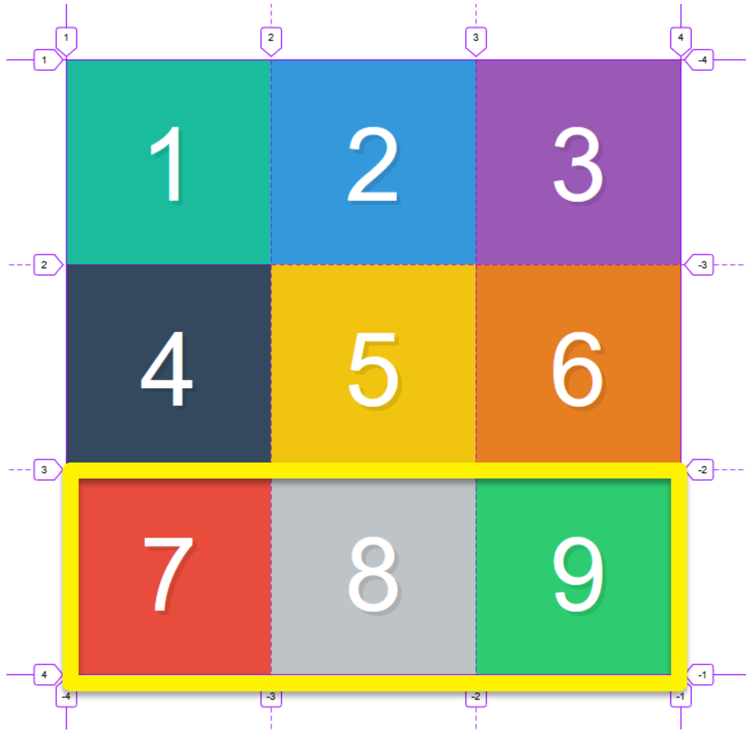
The numbered lines on the right are the grid lines of this 3x3 grid.



Grid Track

The **Grid Track** is the space between two adjacent grid lines. Essentially, the columns and rows of the grid.

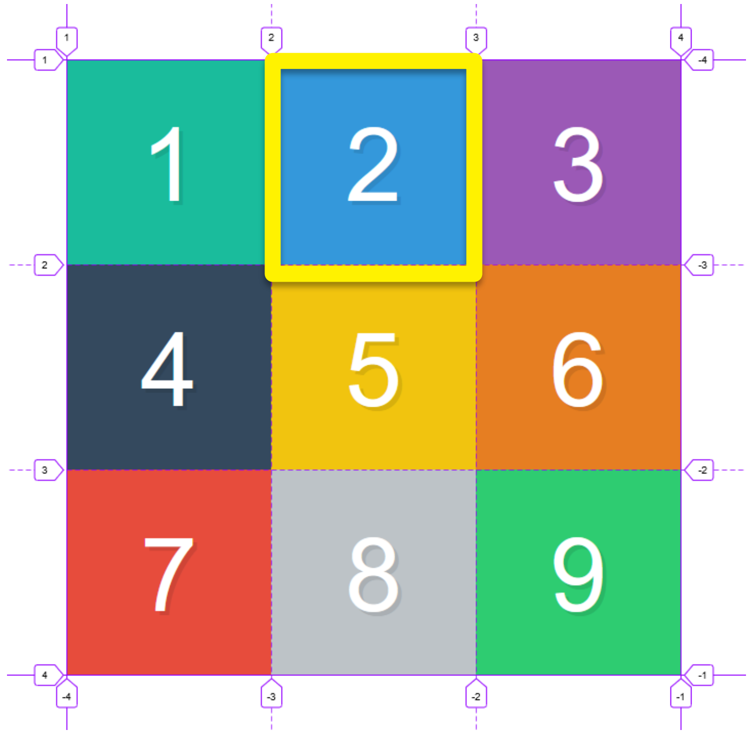
The selected row of 7, 8, 9 is an example of a grid track because it is between row grid lines 3 and 4.



Grid Cell

The **Grid Cell** is the space between two adjacent row grid lines and two adjacent column grid lines. It is a single unit of the grid.

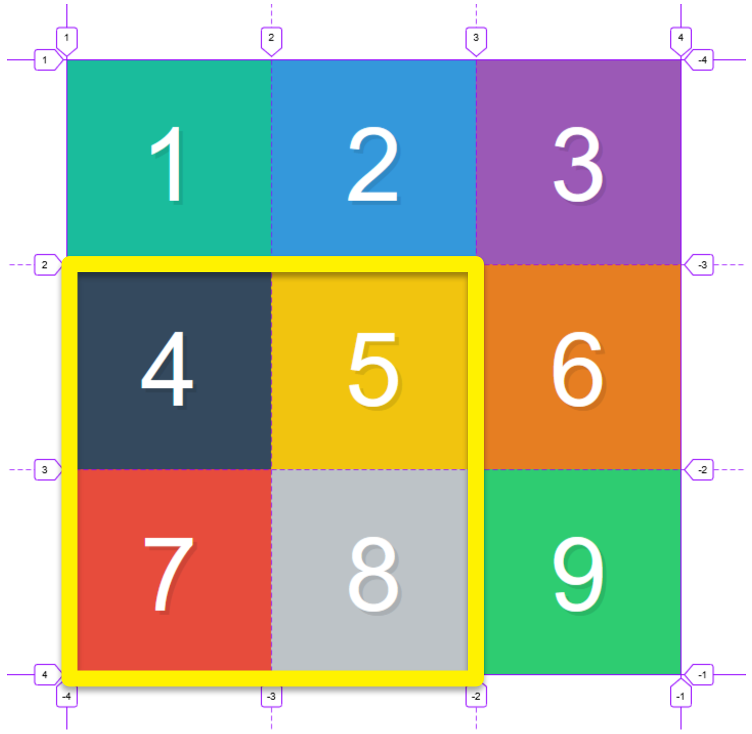
The selected unit of 2 is an example of a grid cell because it is between row grid lines 1 and 2, and column grid lines 2 and 3.



Grid Area

The **Grid Area** is the total space surrounded by four grid lines and can contain any number of grid cells.

The selected grid area is between row grid lines 2 and 4, and column grid lines 1 and 3.



Defining the Grid

Setting `display: grid` on an element is not enough to get a grid layout.

You have to tell the browser the layout for all children of that element. This can be done using these properties:

- grid-template-columns
- grid-template-rows

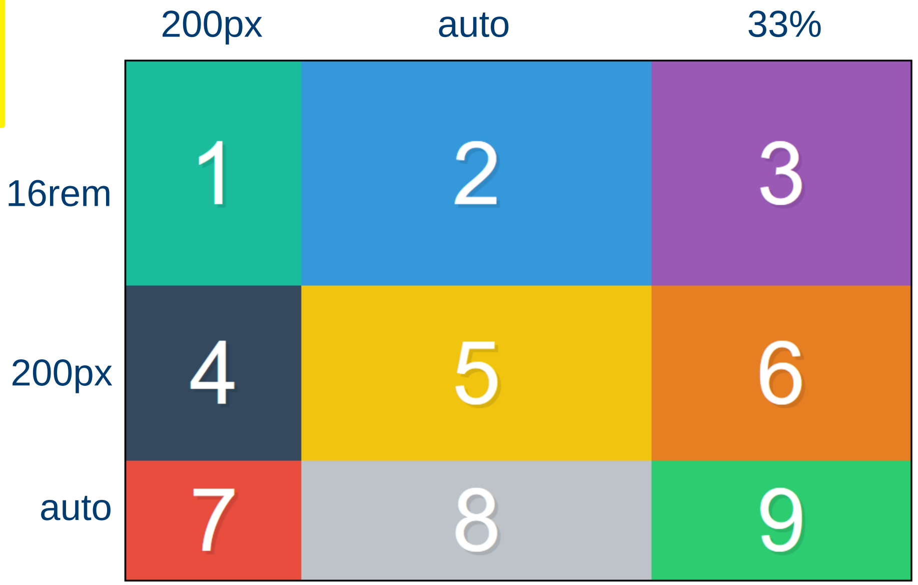
Defining the Grid

```
.container {
	display: grid;
	grid-template-columns: 200px auto 33%;
	grid-template-rows: 16rem 200px auto;
}
```

Each value represents a new column or row.

Values can mix and match CSS units.

The ‘auto’ value will fill available space.



Defining the Grid - Values

In addition to the typical CSS values like pixels, rems, and percentages… `grid-template-columns` and `grid-template-rows` have some unique values.

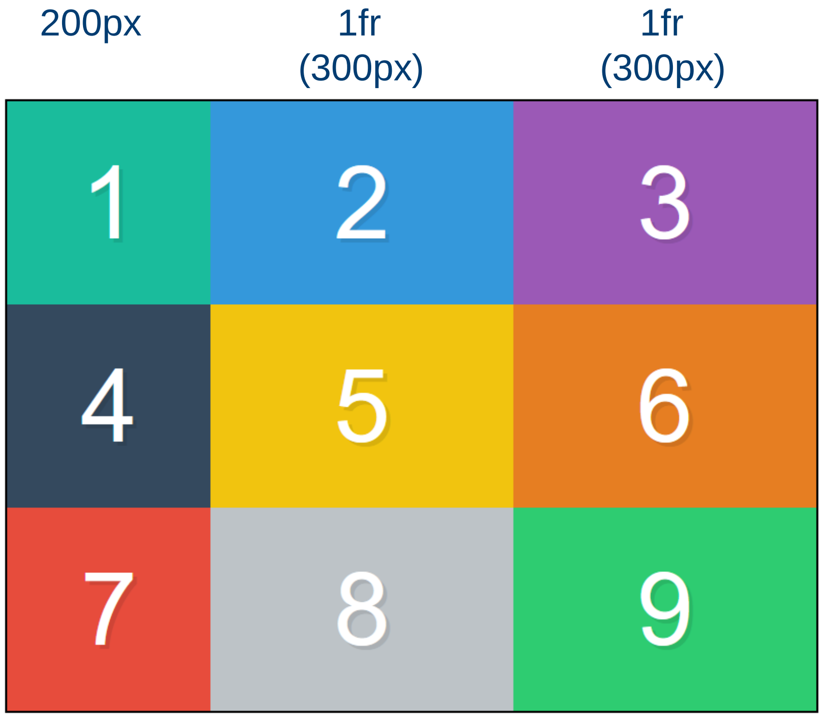
The most commonly used are…

- `fr`
- `repeat()`
- `minmax()`

The fr Unit

The **`fr`** unit stands for the fraction of leftover space in the grid container.

```
.container {
    width: 800px;
    display: grid;
    grid-template-columns: 200px 1fr 1fr;
}
```

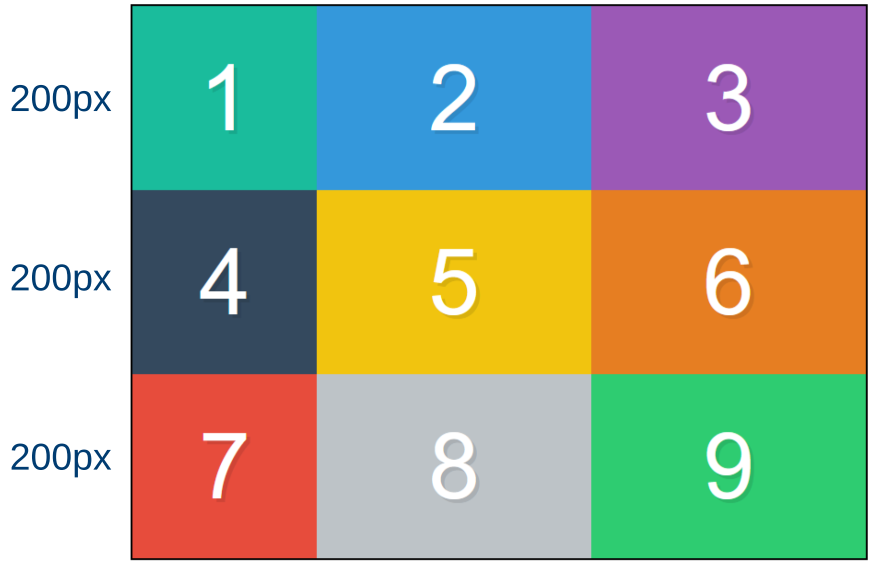


The repeat() Function

The **`repeat()`** function is shorthand for creating multiple items of the same size.

```
.container {
    grid-template-rows: repeat(3, 200px);
}
```

```
.container {
    grid-template-rows: 200px 200px 200px;
}
```



The repeat() Function

The first value in the repeat function is how many times to repeat whatever is after the comma.

You can create multiple rows or columns and repeat them. This will create four rows…

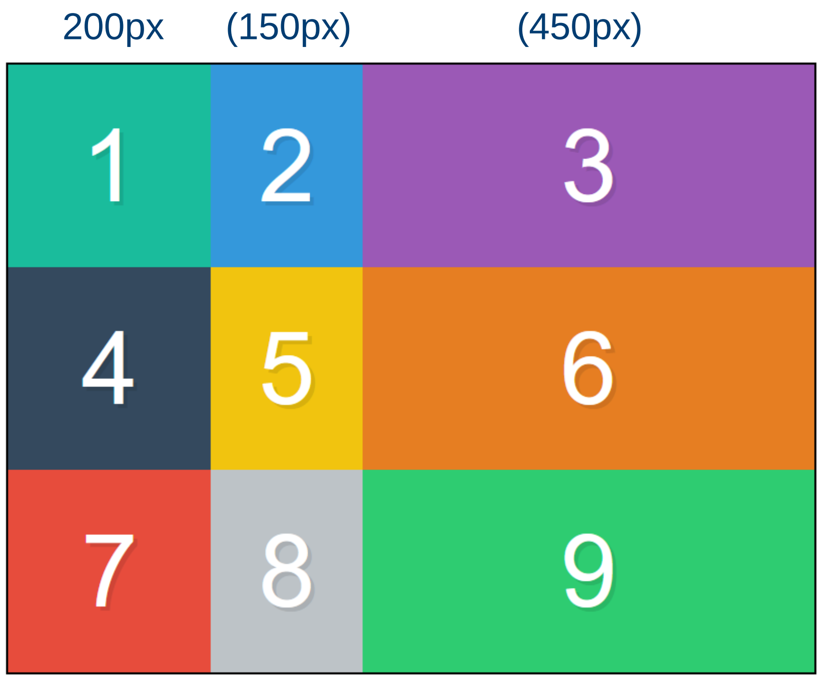
```
    grid-template-rows: repeat(2, 80px 1fr);
```

https://developer.mozilla.org/en-US/docs/Web/CSS/repeat()

The minmax() Function

The **`minmax()`** function is used to set a minimum and maximum value for the track.

```
.container {
     width: 800px;
     display: grid;
     grid-template-columns:
          200px
          minmax(80px, 150px)
          1fr;
}
```



The minmax() Function

The first value in the minmax function is how small the row or column can be and the second value is how large the row or column can be.

The `auto` keyword is handy in either position…

```
    grid-template-columns: 1fr minmax(auto, 600px) 1fr;
```

https://developer.mozilla.org/en-US/docs/Web/CSS/minmax()

Recap

If a Grid is created with three columns, how many vertical grid lines will there be?

2 3 4 6

True or False: Setting display: grid on an element immediately creates a grid layout on the child elements.

Grid Column & Grid Row

The `grid-column` and `grid-row` properties can be set on grid items to determine where on the grid the item appears.

This is done by using the grid line numbers.

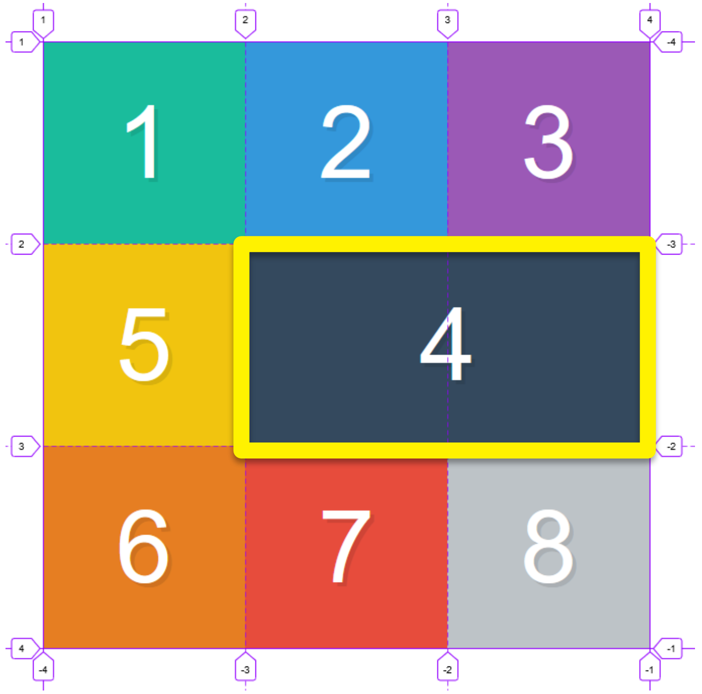
https://developer.mozilla.org/en-US/docs/Web/CSS/grid-column

https://developer.mozilla.org/en-US/docs/Web/CSS/grid-row

Grid Column & Grid Row – Syntax

Box 4 is from row grid line 2 to row grid line 3 and from column grid line 2 to column grid line 4.

```
grid-row: 2 / 3;
grid-column: 2 / 4;
```



Grid Column & Grid Row – Syntax

The first value is the starting line followed by a backlash and then the second value is the ending line.

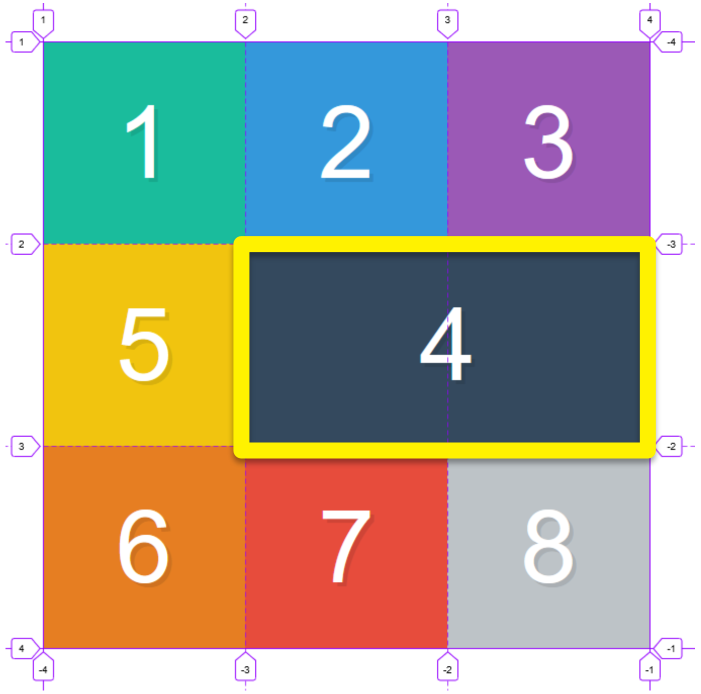
To start from the bottom or right side of the grid, you can use negative numbers.

You can also use the **span** keyword to tell how many rows or columns the grid item should span.

Grid Column & Grid Row – Syntax

This code produces the same results as the previous code but instead of providing the ending column line we say "span 2 columns".

```
grid-row: 2 / 3;
grid-column: 2 / span 2;
```



Overlapping Grid Items

You can use `grid-row` and `grid-column` to have grid items overlap partially or fully.

Once that is done, you can use `z-index` to stack them according to your design.

Auto-Placement in Grid

Grid has methods for dealing with grid items that aren’t explicitly placed in the grid. For example… if you don’t know how many rows you’ll have but want them to all be the same height, use `grid-auto-rows` to define a height for all rows.

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Auto-placement_in_CSS_Grid_Layout

MDN
https://developer.mozilla.org/en-US/docs/Web/CSS/grid-auto-columns
https://developer.mozilla.org/en-US/docs/Web/CSS/grid-auto-rows
https://developer.mozilla.org/en-US/docs/Web/CSS/grid-auto-flow

CSS Tricks
https://css-tricks.com/snippets/css/complete-guide-grid/#prop-grid-auto-columns-rows
https://css-tricks.com/snippets/css/complete-guide-grid/#prop-grid-auto-flow

Object Fit & Object Position

This is **not** specific to CSS Grid but very useful to prevent distorting images when setting heights on them.

Both properties are used on background images and `<img>` elements.

https://developer.mozilla.org/en-US/docs/Web/CSS/object-fit

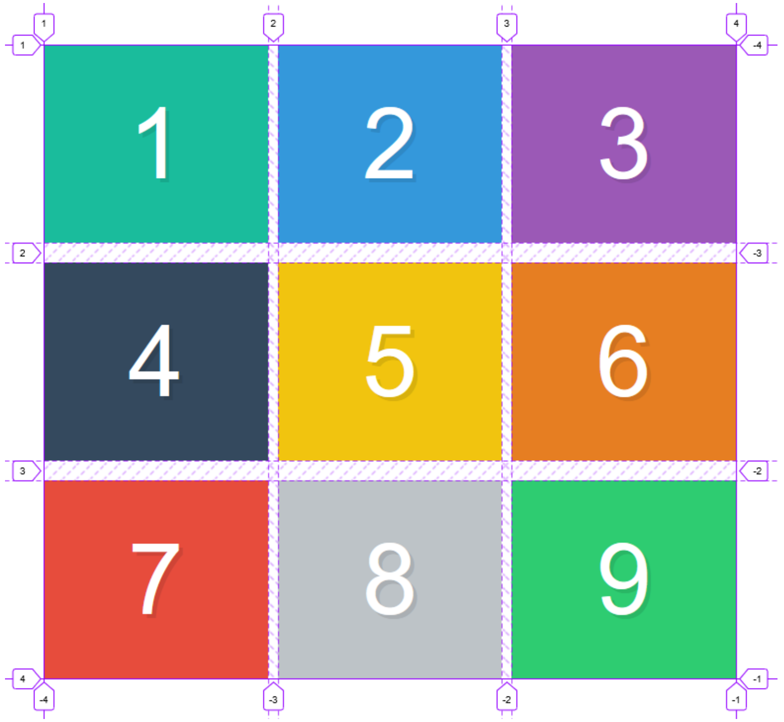
https://developer.mozilla.org/en-US/docs/Web/CSS/object-position

Gap

The grid **Gap** is the size of the grid lines and defined on the grid container.

The first value sets the row gap and the second value sets the column gap. For example:

```
gap: 20px 10px;
```



Grid Alignment Properties

Grid and Flexbox use similar alignment properties.

These properties are set on the grid container:

- `justify-items`
- `justify-content`
- `align-items`
- `align-content`

These properties are set on the grid item:

- `justify-self`
- `align-self`

Grid Container Alignment

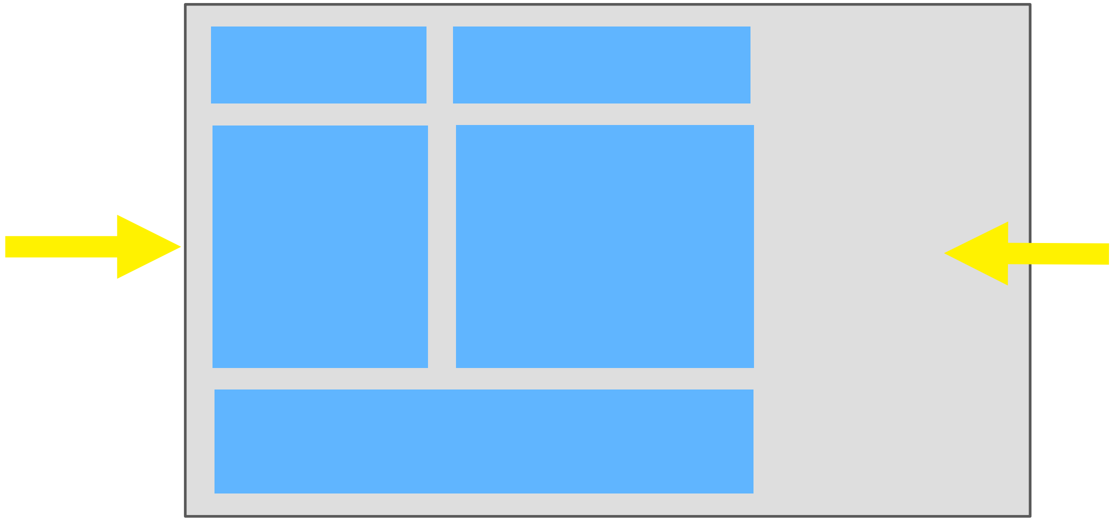
If your grid is smaller than the grid container, you can use these two properties to align the grid within the container horizontally and vertically respectively.

```
justify-content        align-content
```

You probably won’t use these much. These are usually only needed when using fixed sizes like pixels and rems.

Justify Content

Grid Container



Empty space within the grid container

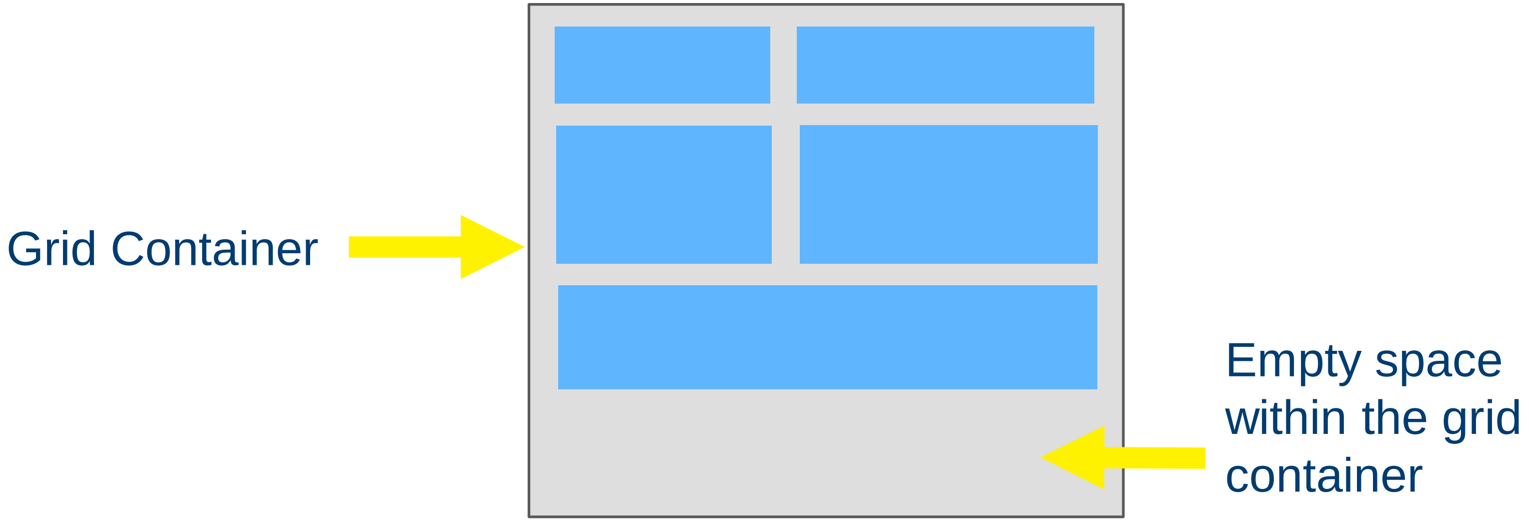
https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content
https://css-tricks.com/snippets/css/complete-guide-grid/#prop-justify-content

Justify Content

The justify-content property has the following possible values:

- **start** – aligns the grid to be flush with the start edge of the grid container
- **end** – aligns the grid to be flush with the end edge of the grid container
- **center** – aligns the grid in the center of the grid container
- **stretch** – resizes the grid items to allow the grid to fill the full width of the grid container
- **space-around** – places an even amount of space between each grid item, with half-sized spaces on the far ends
- **space-between** – places an even amount of space between each grid item, with no space at the far ends
- **space-evenly** – places an even amount of space between each grid item, including the far ends

Align Content



https://developer.mozilla.org/en-US/docs/Web/CSS/align-content
https://css-tricks.com/snippets/css/complete-guide-grid/#prop-align-content

Align Content

The align-content property has the following possible values:

- **start** – aligns the grid to be flush with the start edge of the grid container
- **end** – aligns the grid to be flush with the end edge of the grid container
- **center** – aligns the grid in the center of the grid container
- **stretch** – resizes the grid items to allow the grid to fill the full height of the grid container
- **space-around** – places an even amount of space between each grid item, with half-sized spaces on the far ends
- **space-between** – places an even amount of space between each grid item, with no space at the far ends
- **space-evenly** – places an even amount of space between each grid item, including the far ends

Place Content

The place-content property is a shorthand to set align-content and justify-content at once.

It has the same possible values.

https://developer.mozilla.org/en-US/docs/Web/CSS/place-content
https://css-tricks.com/snippets/css/complete-guide-grid/#place-content

Grid Items Alignment

These two values determine how the grid items are positioned between the grid lines for the entire grid container.

```
justify-items        align-items
```

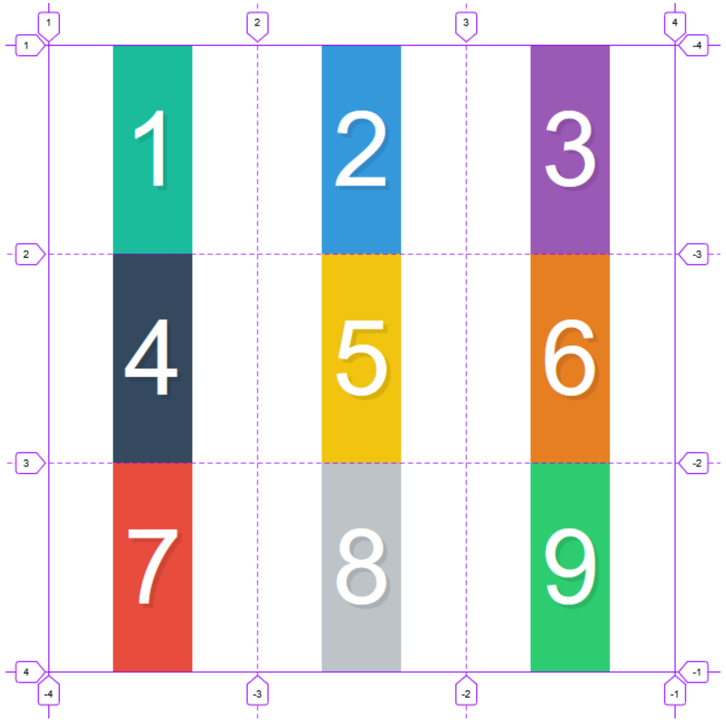
Justify-items handles the row alignments, align-items handles the column alignments.

Justify Items

In this example we have fixed grid item sizes (200px) but the content is smaller than that.

With `justify-items: center` there is whitespace now in the rows.

```
.container {
    display: grid;
    grid-template-columns: 200px 200px 200px;
    grid-template-rows: 200px 200px 200px;
    justify-items: center;
}
```



Justify Items

The justify-items property has the following possible 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)

https://developer.mozilla.org/en-US/docs/Web/CSS/justify-items

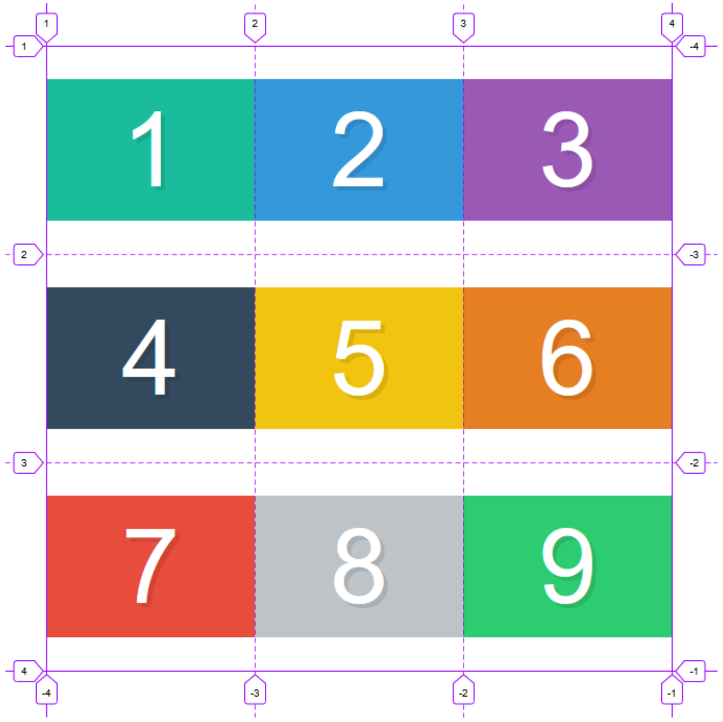
https://css-tricks.com/snippets/css/complete-guide-grid/#prop-justify-items

Align Items

In this example we have fixed grid item sizes (200px) but the content is smaller than that.

With `align-items: center` there is whitespace now in the columns.

```
.container {
    display: grid;
    grid-template-columns: 200px 200px 200px;
    grid-template-rows: 200px 200px 200px;
    align-items: center;
}
```



Align Items

The align-items property has the following possible 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)

https://developer.mozilla.org/en-US/docs/Web/CSS/align-items
https://css-tricks.com/snippets/css/complete-guide-grid/#prop-align-items

Place Items

The place-items property is a shorthand to set align-items and justify-items at once. It has the same possible 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)

https://developer.mozilla.org/en-US/docs/Web/CSS/place-items

https://css-tricks.com/snippets/css/complete-guide-grid/#place-items

Single Grid Item Alignment

These two values determine how a single grid item is positioned between the grid lines.

`justify-self`   `align-self`

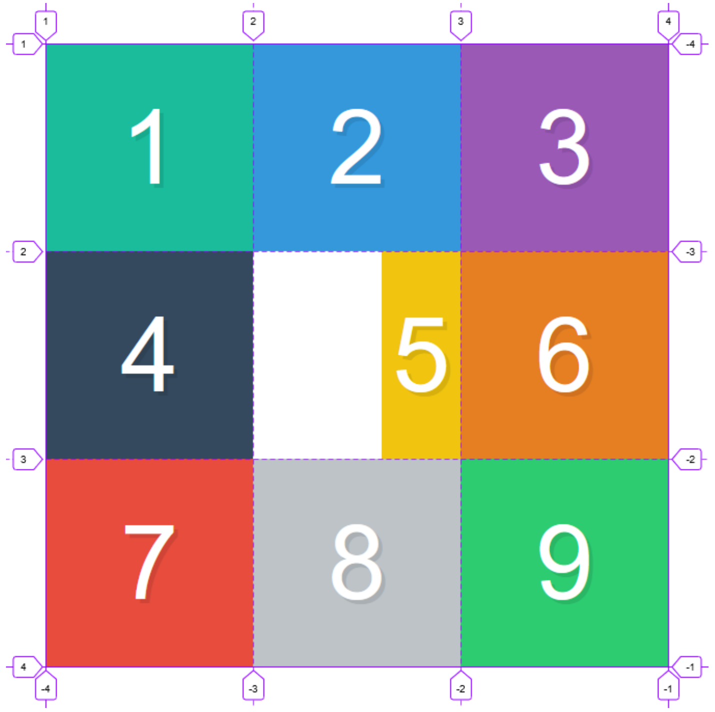
Justify-self handles the row alignment, align-self handles the column alignment. They have the same values as `justify-items` and `align-items`.

Justify Self

In this example we have fixed grid item sizes (200px) but the content is smaller than that.

With `justify-self: end;` there is whitespace now in the row.

```
.box-5 {
    justify-self: end;
}
```



Justify Self

The justify-self property has the following possible values:

- **start** – aligns the grid item to be flush with the start edge of the cell
- **end** – aligns the grid item to be flush with the end edge of the cell
- **center** – aligns the grid item in the center of the cell
- **stretch** – fills the whole width of the cell (this is the default)

https://developer.mozilla.org/en-US/docs/Web/CSS/justify-self

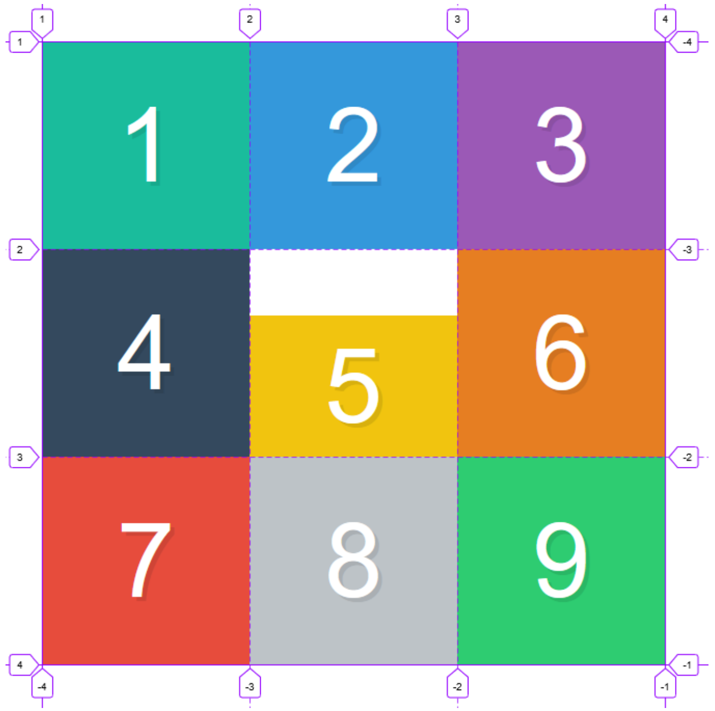
https://css-tricks.com/snippets/css/complete-guide-grid/#prop-justify-self

Align Self

In this example we have fixed grid item sizes (200px) but the content is smaller than that.

With `align-self: end;` there is whitespace now in the column.

```
.box-5 {
     align-self: end;
}
```



Align Self

The align-self property has the following possible values:

- **start** – aligns the grid item to be flush with the start edge of the cell
- **end** – aligns the grid item to be flush with the end edge of the cell
- **center** – aligns the grid item in the center of the cell
- **stretch** – fills the whole height of the cell (this is the default)

https://developer.mozilla.org/en-US/docs/Web/CSS/align-self
https://css-tricks.com/snippets/css/complete-guide-grid/#prop-align-self

Place Self

The place-self property is a shorthand to set align-self and justify-self at once. It has the same possible values:

- **start** – aligns the grid item to be flush with the start edge of the cell
- **end** – aligns the grid item to be flush with the end edge of the cell
- **center** – aligns the grid item in the center of the cell
- **stretch** – fills the whole height of the cell (this is the default)

https://developer.mozilla.org/en-US/docs/Web/CSS/place-self
https://css-tricks.com/snippets/css/complete-guide-grid/#place-self

Order

The `order` property can be used on grid items in the same way that it can be used on flex items.

The default order value is 0.

It can accept positive or negative integers.

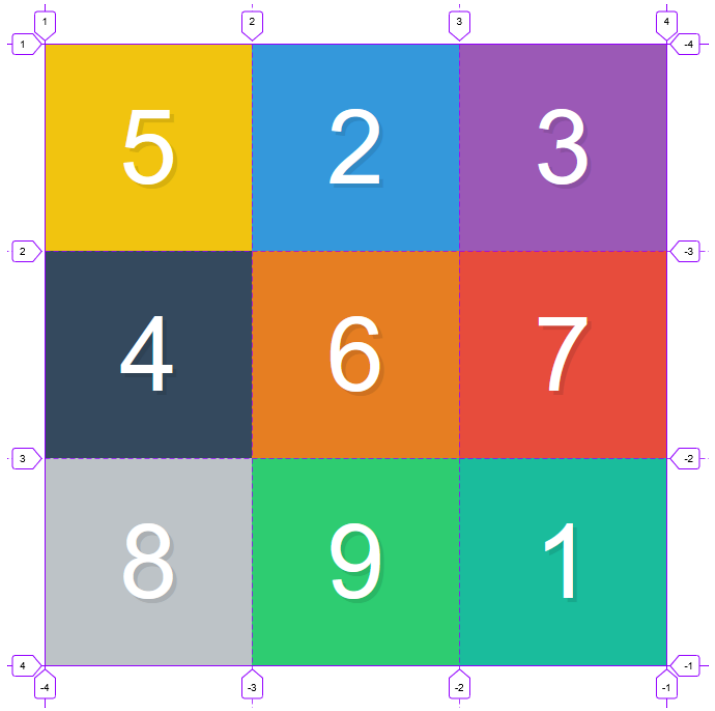
https://developer.mozilla.org/en-US/docs/Web/CSS/order

Order

Because the default value of order is 0, using 1 and -1 can quickly move items to the beginning and end of a grid.

```
.box-1 {
     order: 1;
}

.box-5 {
     order: -1;
}
```



Grid Template Areas

Grid Template Areas can be used in combination with Grid Template Columns and Grid Template Rows.

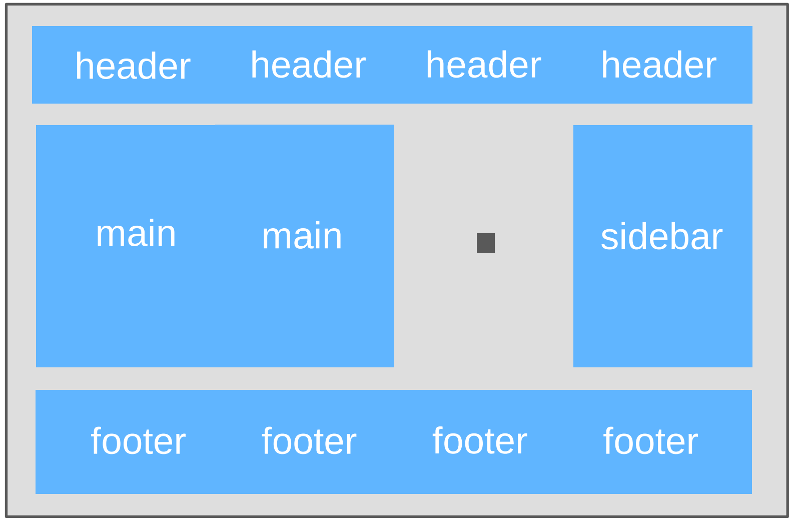
They are very handy for making advanced layouts more intuitive in your CSS so you don’t have to keep track of grid lines.

https://developer.mozilla.org/en-US/docs/Web/CSS/grid-template-areas

Grid Template Areas

The `grid-template-areas` property is used on the grid container.

The `grid-area` property is used on the grid items.



Grid Template Areas

```
.site-wrapper {
    display: grid;
    grid-template-columns: 100px 100px 100px 100px;
    grid-template-rows: auto;
    grid-template-areas:
        "header header header header"
        "main   main   .      sidebar"
        "footer footer footer footer";
}
.site-header    { grid-area: header;  }
.site-main      { grid-area: main;    }
.site-sidebar   { grid-area: sidebar; }
.site-footer    { grid-area: footer;  }
```

The text should describe the visual layout of the page.

A dot indicates an empty grid cell.

Assignment #6

Assignment #6

- Please refer to Assignment #6 in the Learning Hub.
- To submit the assignment, you can do **one** of these:
  - Have me check your assignment in class before 4pm.
  - Zip today's **folder** and upload it to the Learning Hub before next class.
- If you have questions or need guidance, just ask!

Resources

CSS Grid (MDN - Web)
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout

CSS Grid (MDN - Learn)
https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids

CSS Grid (CSS-Tricks)
https://css-tricks.com/snippets/css/complete-guide-grid/

Tools

CSS Grid Playground (Mozilla)
https://mozilladevelopers.github.io/playground/css-grid

Grid by Example
https://gridbyexample.com/examples/

CSS Grid Cheat Sheet
https://alialaa.github.io/css-grid-cheat-sheet/

Games

Grid Critters
https://gridcritters.com/

Grid Garden
https://cssgridgarden.com/

Grid Attack
https://codingfantasy.com/games/css-grid-attack

Video Tutorial

LinkedIn Learning – CSS: Advanced Layouts with Grid
https://www.linkedin.com/learning/css-advanced-layouts-with-grid/

QUESTIONS & ANSWERS