What does @media only screen and (min-width: 700px) mean?

The @media rule is used for CSS media queries, which allow you to apply styles based on the screen size, device type, or other conditions.

only screen → This ensures that the styles inside the query apply only to screen devices (ignoring print or other media types).

(min-width: 700px) → This means that the styles inside this block will only be applied when the screen width is 700 pixels or more.

/\* For small screens (up to 600px) \*/

@media (max-width: 600px) {

body {

background-color: lightblue;

}

}

/\* For medium screens (from 601px to 1024px) \*/

@media (min-width: 601px) and (max-width: 1024px) {

body {

background-color: lightgreen;

}

}

/\* For large screens (above 1024px) \*/

@media (min-width: 1025px) {

body {

background-color: lightcoral;

}

}

Small screens (≤600px) → light blue background

Medium screens (601px - 1024px) → light green background

Large screens (>1024px) → light coral background

What does it mean by

.blog-item:hover p {

  color: var(--color-1);

}

.blog-item:hover p targets the <p> element inside .blog-item only when .blog-item is hovered over.

If there were other <p> elements in your code (outside .blog-item), they would remain unchanged unless they were explicitly targeted by another CSS rule.

What do all the properties mentioned here mean, and what is their purpose

.container .left {

flex: .7;

display: grid;

grid-template: 1fr 1fr / 1fr 1fr ;

grid-gap: 20px;

}

**flex: 0.7**; → This means the .left container takes up 70% of the available space in its flex container. The remaining space is given to the other sibling elements based on their flex values.

If you remove flex: 0.3; from .right, then:

Flex Items Default to auto Sizing

The .right container will only take up as much space as needed by its content.

It won't stretch unless there's extra space available.

Left Section (.left) Will Expand

Since .left has flex: 0.7;, it will take 70% of the space.

The remaining space will be available for .right, but it will only take what it needs instead of expanding to 30%.

**display: grid**; → This makes .left a grid container, allowing child elements to be placed in a structured grid format.

**grid-template**: 1fr 1fr / 1fr 1fr; → Defines a grid with two rows (1fr 1fr) and two columns (1fr 1fr). Each fraction (fr) represents a portion of available space, so all rows and columns are equal in size.

**grid-template: 1fr 1fr / 1fr 1fr;**

This property is shorthand for defining both rows and columns in a CSS Grid layout. Let's separate it:

Rows: 1fr 1fr → This means the grid has two rows, each taking an equal portion (1fr each) of the available height.

Columns: 1fr 1fr → This means the grid has two columns, each taking an equal portion (1fr each) of the available width.

grid-template: 1fr / 1fr 1fr 1fr;

This creates one row but three columns

When you change the grid-template to 1fr / 1fr, it doesn't affect the layout because of the implicit grid behavior in CSS Grid.

1fr / 1fr; defines one row and one column.

However, since you have more than one item inside .left, CSS Grid automatically places additional items in new implicit rows.

When you reduce grid-template to a single column, Grid automatically adds rows for extra items.