**GRID** **31st Jan**

1. What is the Grid?

Ans- The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

A grid can be defined as an intersecting set of vertical and horizontal lines. CSS Grid layout separates a page into major sections. Grid property offers a grid-based layout system having rows and columns. It makes the designing of web pages easy without positioning and floating. The grid layout gives us a way to create grid structures depicted in CSS, not in HTML.

Similar to the table, it enables a user to align the elements into rows and columns. But compare to tables, it is easy to design layout with the CSS grid. We can define columns and rows on the grid by using grid-template-rows and grid-template-columns properties.

A grid container can be created by declaring the display: grid or display: inline-grid on an element. Grid container contains the items of a grid that are placed inside the rows and columns.

A Grid Layout must have a parent element with the *display* property set **to *grid* or *inline-grid***.

Direct child element(s) of the grid container automatically becomes grid items.

With CSS Grid, you can define both the rows and columns of the grid using the **grid-template-rows** and **grid-template-columns** properties. These properties allow you to specify the size and structure of the grid tracks, which are the individual cells or areas within the grid.

Once the grid is established, you can place elements, known as grid items, within the grid using properties such as **grid-row** and **grid-column**. These properties determine the starting and ending positions of the grid items within the grid tracks, allowing you to define their position and span across multiple rows or columns.

CSS Grid provides powerful alignment capabilities, allowing you to align grid items along both the horizontal and vertical axes. You can control the alignment of items within their grid cells using properties such as **justify-self** and **align-self**, or align items within the grid container using properties like **justify-items** and **align-items**.

1. What is the difference between Flex and Grid?

Ans- Flexbox and Grid are two CSS layout models that serve different purposes and have different capabilities. Here are the key differences between Flexbox and Grid:

1. **Purpose and Layout Approach**:
   1. Flexbox: Flexbox, or Flexible Box Layout, is primarily designed for creating one-dimensional layouts, either horizontally or vertically. It allows you to arrange elements in a row or column and control their size, order, and alignment within that single dimension.
   2. Grid: CSS Grid Layout, commonly referred to as Grid, is designed for creating two-dimensional layouts. It provides a grid-based system where you can define both rows and columns and place elements anywhere within the grid. It offers precise control over the placement, size, and alignment of elements in both dimensions.
2. **Dimensionality:**
   1. Flexbox: Flexbox is a one-dimensional layout model. It focuses on arranging elements along a single row or column. It is ideal for creating flexible and responsive layouts that adapt to different screen sizes and content.
   2. Grid: Grid is a two-dimensional layout model. It allows you to create complex layouts with rows and columns, enabling more extensive control over the positioning and alignment of elements in both the horizontal and vertical dimensions. It is suitable for creating grid-like structures, such as card layouts or complex web page designs.
3. **Alignment and Ordering:**
   1. Flexbox: Flexbox provides powerful alignment capabilities, allowing you to align elements along the main axis or cross axis, control their order, and distribute remaining space among them. It excels at centering elements, creating equal height columns, and distributing space dynamically.
   2. Grid: Grid offers robust alignment capabilities as well, but it extends beyond a single axis. It enables precise control over the placement and alignment of elements within individual cells of the grid. Grid allows you to align elements both horizontally and vertically and control their span across multiple cells.
4. **Nesting:**
   1. Flexbox: Flexbox allows for easy nesting of flex containers within other flex containers. This feature helps create more complex layouts where child flex containers can have their own flex behavior independently.
   2. Grid: Grid also allows for nesting grid containers within other grid containers. This enables the creation of complex grid structures where different sections of the layout can have their own individual grid systems.
5. How can you define rows and columns for your grid?

Ans- The grid-template-rows property specifies the number (and the heights) of the rows in a grid layout.

The values are a space-separated list, where each value specifies the height of the respective row.

Example:

Specify the row-size (height):

.grid-container {  
  display: grid;  
  grid-template-rows: 100px 300px;   
}

**CSS Syntax**

grid-template-rows: none|auto|max-content|min-content|*length*;

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1.none | No size is set. Rows are created if needed |
| 2.auto | The size of the rows is determined by the size of the container, and on the size of the content of the items in the row |
| 3.max-content | Sets the size of each row to depend on the largest item in the row |
| 4.min-content | Sets the size of each row to depend on the smallest item in the row |

The grid-template-columns property specifies the number (and the widths) of columns in a grid layout.

The values are a space separated list, where each value specifies the size of the respective column.

Example:

Make a 4 columns grid container:

.grid-container {  
  display: grid;  
  grid-template-columns: 100px 200px 100px auto;  
}

**CSS Syntax**

grid-template-columns: none|auto|max-content|min-content|*length*;

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1. none | Default value. Columns are created if needed |
| 1. auto | The size of the columns is determined by the size of the container and on the size of the content of the items in the column |
| 1. max-content | Default value. Columns are created if needed |
| 1. min-content | Sets the size of each column to depend on the smallest item in the column |
| 1. length | Sets the size of the columns, by using a legal length value |

1. List any two properties of the grid item and grid container.

Ans- **Properties of Grid Item:**

1. grid-column: This property specifies the starting and ending positions of the grid item within the grid columns. It allows you to define the size and position of the item horizontally.

Example :

.item {

grid-column: 1 / 3; /\* item spans from column line 1 to column line 3 \*/

}

1. grid-row: This property determines the starting and ending positions of the grid item within the grid rows. It defines the size and position of the item vertically.

Example :

.item {

grid-row: 2 / 4; /\* item spans from row line 2 to row line 4 \*/

}

1. justify-self: This property aligns the content of a grid item along the horizontal axis within its grid cell. It allows you to control the self-alignment of a single grid item.

Example :

.item {

justify-self: center; /\* item is horizontally centered within its grid cell \*/

}

The justify-self property accepts various values such as start, end, center, stretch, baseline, and auto, among others. These values control the horizontal alignment of the grid item content within its grid cell.

By using justify-self, you can easily position individual grid items horizontally within the grid container, allowing for more precise control over the layout and alignment of the content.

**Properties of Grid Container:**

1. display: This property sets the container element as a grid container. It enables the grid layout for the container and establishes a new grid formatting context.

Example:

.container {

display: grid; /\* container becomes a grid container \*/

}

1. grid-template-columns: This property defines the number and size of the columns in the grid container. It allows you to specify the width of each column individually or by using flexible units like percentages or fractions.

Example:

.container {

grid-template-columns: 1fr 2fr 1fr;

/\* three columns with the first and third column having equal width and the second column being twice as wide \*/

}

Please note that these are just two examples of properties related to grid items and grid containers. There are several other properties available for customization in CSS Grid Layout.

1. grid-gap: This property sets the size of the gap between grid rows and columns. It defines the spacing between grid tracks, allowing you to create space between grid items.

Example:

.container {

grid-gap: 10px; /\* 10 pixels of space between grid rows and columns \*/

}

The grid-gap property can also be specified separately for row and column gaps using grid-row-gap and grid-column-gap properties, respectively. Additionally, you can use different units like percentages, ems, rems, etc., to define the size of the gaps.

By using grid-gap or its related properties, you can easily control the spacing between grid items, providing better visual separation and alignment within the grid container.