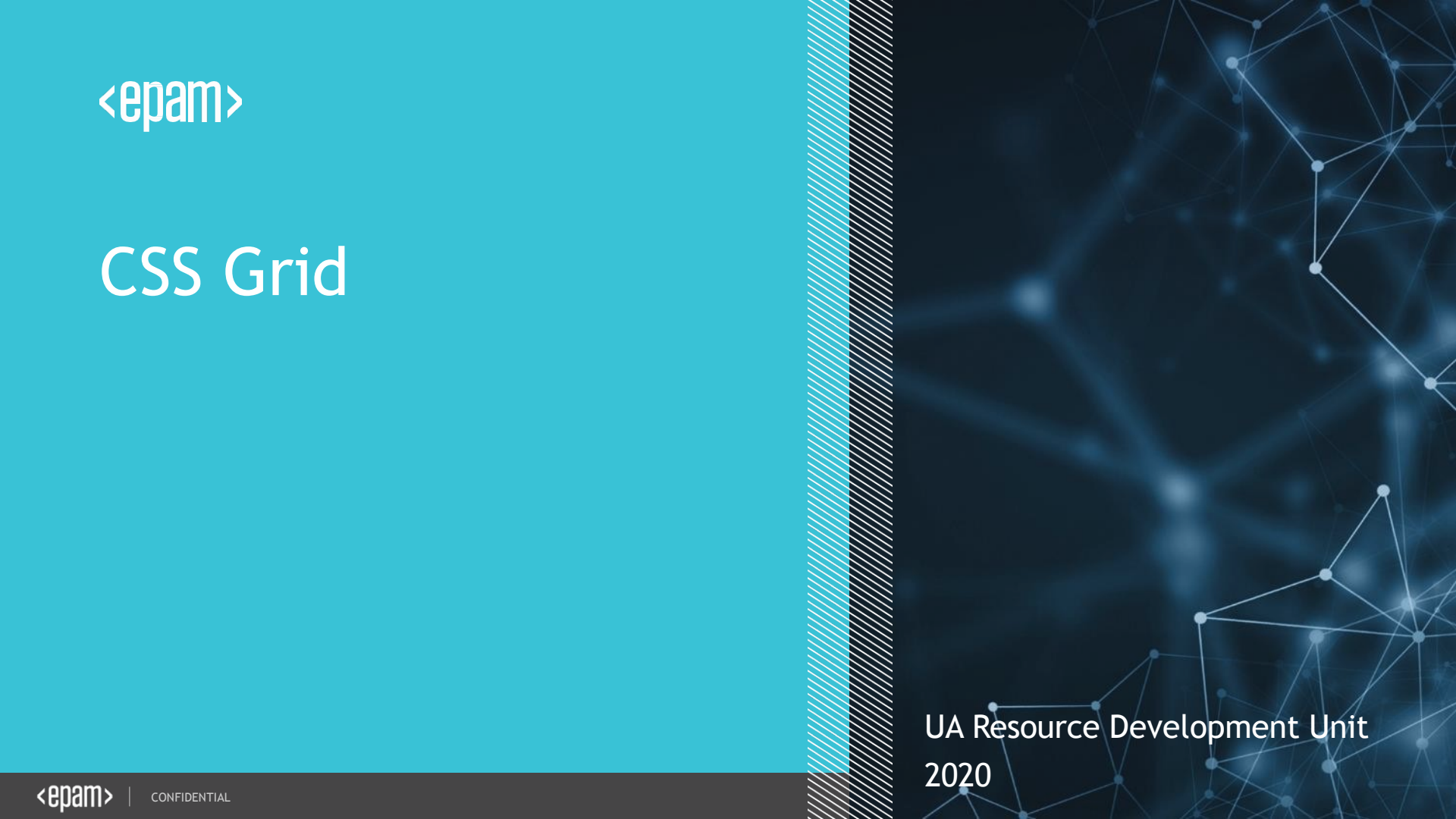
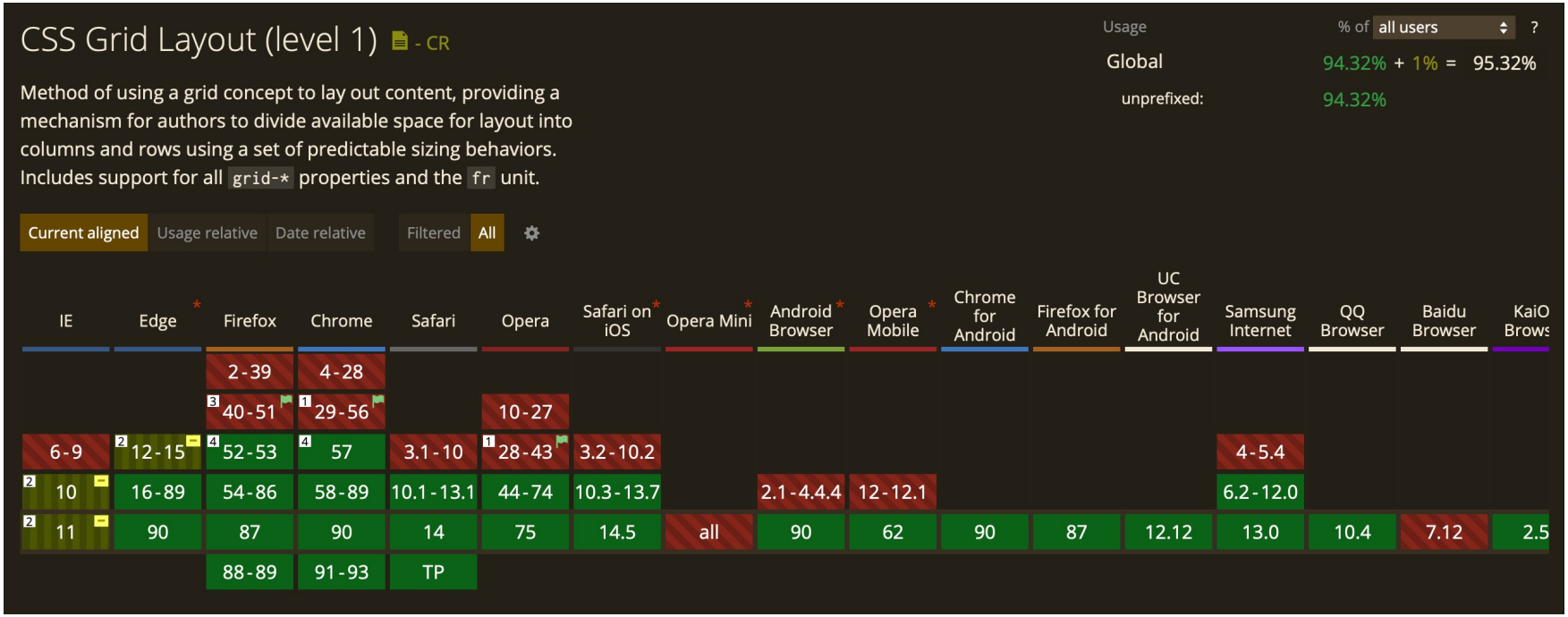

epam
CSS Grid
UA Resource Development Unit
2020
epam
CONFIDENTIAL

AGENDA

1. **Grid Lines**
2. **Grid Tracks**
3. **Grid Areas**
4. Grid Alignment
5. The Auto-placement Algorithm

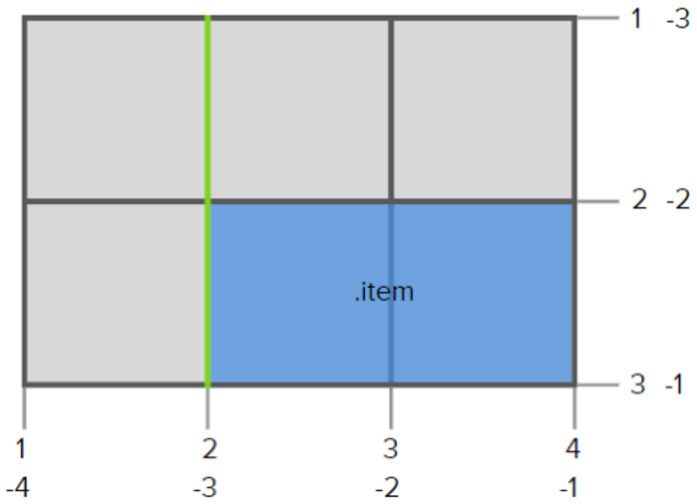
Browser Support for CSS Grid



The Grid Model

- Grids and Grid Lines

row grid line

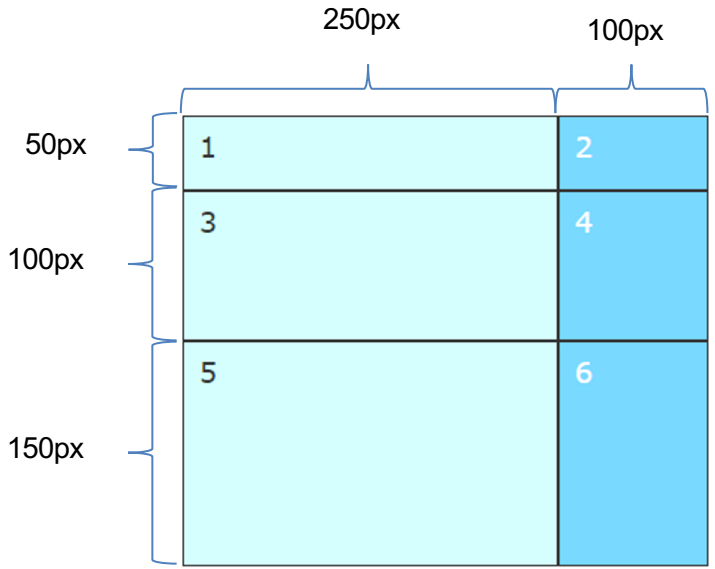


column grid line

```
.grid-container {
  display: grid;
  grid-template-columns: 150px 150px 150px;
  grid-template-rows: 150px 150px;
}

.item {
  grid-column-start: 2;
  grid-column-end: 4;
  grid-row-start: 2;
  grid-row-end: 3;
}
```

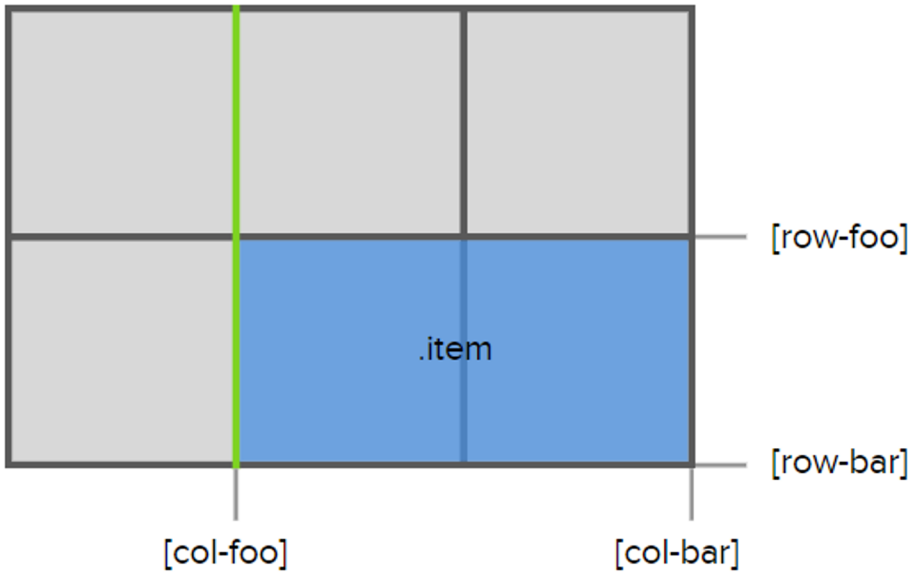
```
div#outer {
    display: grid;
    grid-template-columns: 250px 100px;
    grid-template-rows: 50px 100px 150px;
}
```

explicitly defines both the rows and columns of the grid



The Grid Model

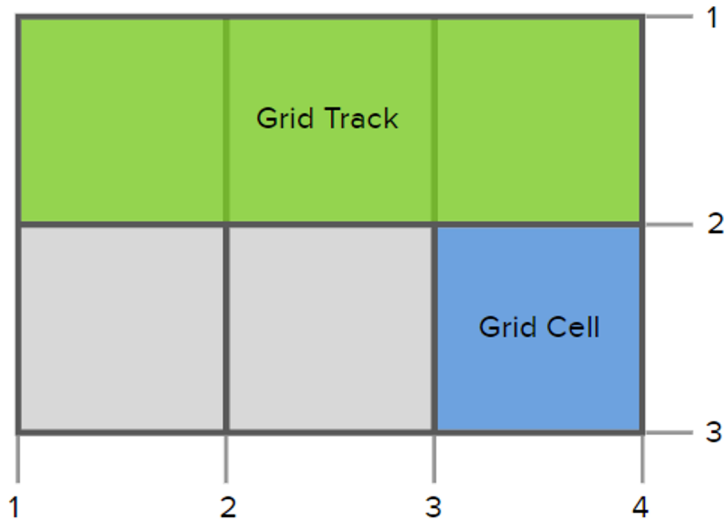
- Grids and Grid Lines




```
.grid-container {
  display: grid;
  grid-template-columns: 150px [col-foo] 150px 150px [col-bar];
  grid-template-rows: 150px [row-foo] 150px [row-bar];
}

.item {
  grid-column-start: col-foo;
  grid-column-end: col-bar;
  grid-row-start: row-foo;
  grid-row-end: row-bar;
}
```

The Grid Model

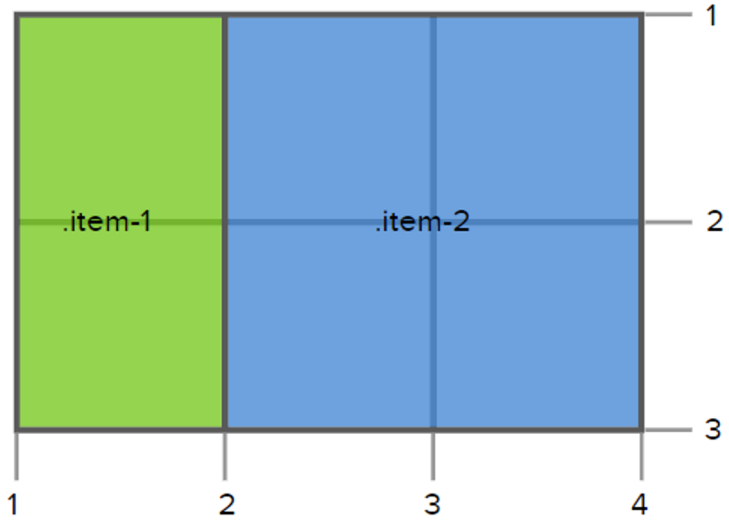
- Grid Tracks




```
.grid-container {
  display: grid;
  grid-template-columns: 150px 150px 150px; /* three columns  */
  grid-template-rows: 150px 150px; /* two rows  */
}
```

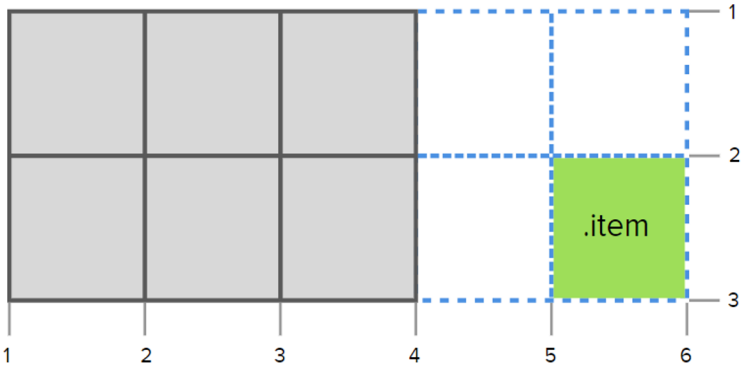
The Grid Model

- Grid Areas



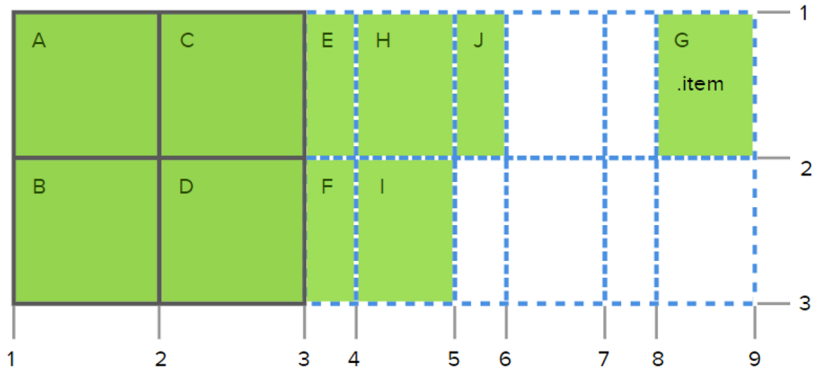

```
.grid-container {
  display: grid;
  grid-template-columns: 150px 150px 150px; /*
three columns  */
  grid-template-rows: 150px 150px; /* two rows  */
  grid-template-areas: "a b b"
                       "a b b";
}
.item-1 {
  grid-area: a;
}
.item-2 {
  grid-area: b;
}
```

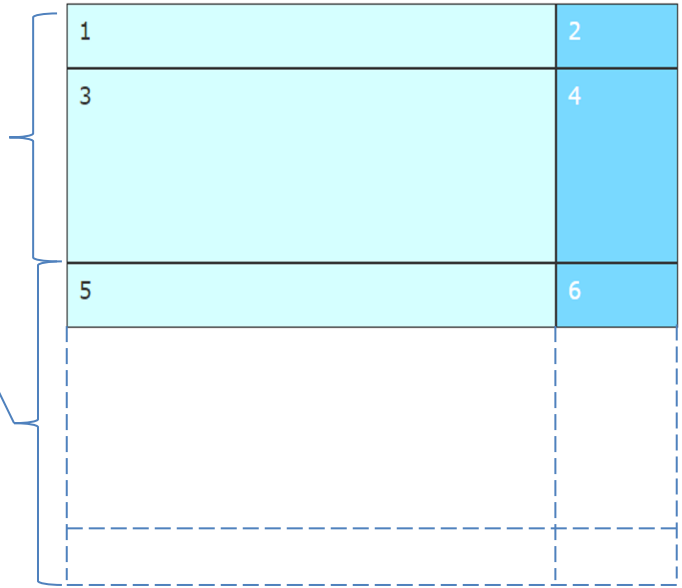
Explicit Grid and Implicit Grid




```
.grid {
  display: grid;
  grid-template-columns: 150px 150px;
  grid-template-rows: 150px 150px;
  grid-auto-columns: 50px 75px;
  grid-auto-flow: column;
}

.item {
  grid-column: 8;
}
```

```
.grid-container {
  display: grid;
  grid-template-columns: 150px 150px 150px; /* three columns  */
  grid-template-rows: 150px 150px; /* two rows  */
}
.item {
  grid-column: 5 / 6; /* grid-column-start: 5; grid-column-end: 6;  */
  grid-row:  2 / 3; /* grid-row-start: 2; grid-row-end: 3;  */
}
```





as additional rows are needed, they continue that pattern

```
div#outer {
    display: grid;
    grid-template-columns: 4fr 1fr;
    grid-auto-rows: 50px 150px;
}
```

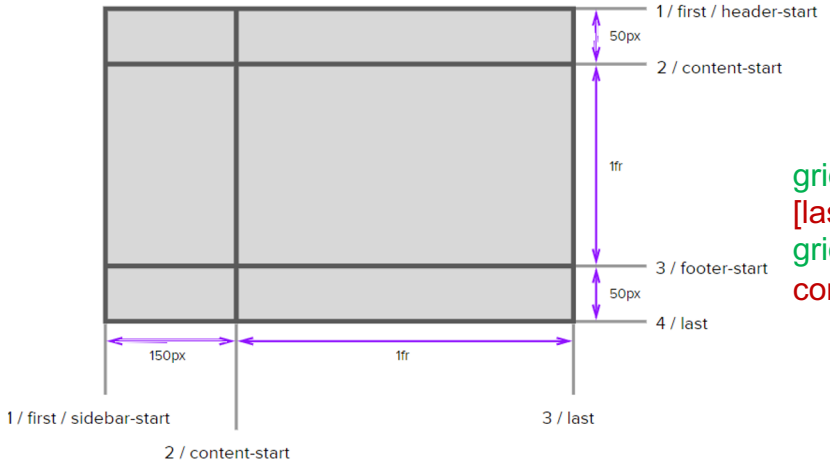
row heights follow the pattern 50px followed by 150px

- The grid-template-columns and grid-template-rows Properties

grid-template-columns: none | <track-list> | <auto-track-list>
grid-template-rows: none | <track-list> | <auto-track-list>

<track-list>

[<line-names>? [<track-size> | <track-repeat>]]+ <line-names>?



grid-template-columns: [first sidebar-start] 150px [content-start] 1fr [last];
grid-template-rows: [first header-start] 50px [content-start] fit-content(300px) 1fr; [footer-start] 50px [last];

<auto-track-list>

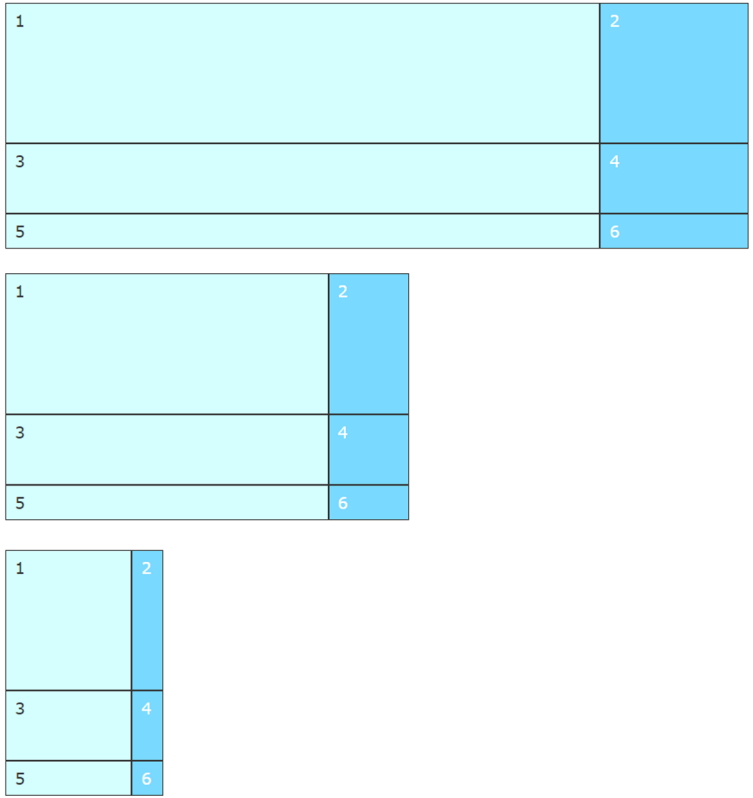
[<line-names>? [<fixed-size> | <fixed-repeat>]]* <line-names>? <auto-repeat>

Fraction

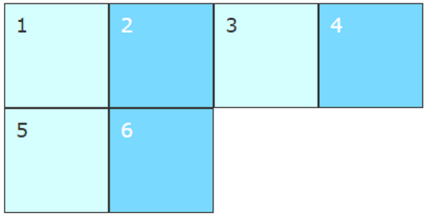
grid columns displayed in a proportion of 4:1

```
div#outer {
    display: grid;
    grid-template-columns: 4fr 1fr;
    grid-template-rows: 4fr 2fr 1fr;
}
```

grid rows displayed in a proportion of 4:2:1

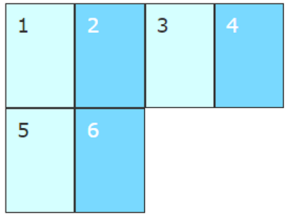


Repeat



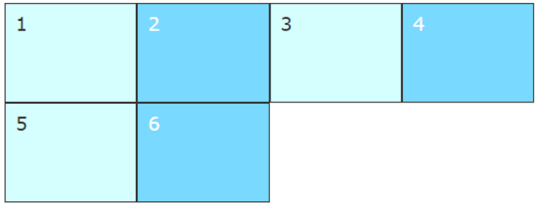

```
div#outer {
    display: grid;
    grid-template-columns: repeat(4, 100px);
    grid-auto-rows: 100px;
}
```

repeat 4 columns of 100px width




```
div#outer {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-auto-rows: 100px;
}
```

repeat 4 equal-width columns




```
div#outer {
    display: grid;
    grid-template-columns: repeat(4, minmax(100px, 1fr));
    grid-auto-rows: 100px;
}
```

repeat 4 equal-width columns with a minimum width of 100px

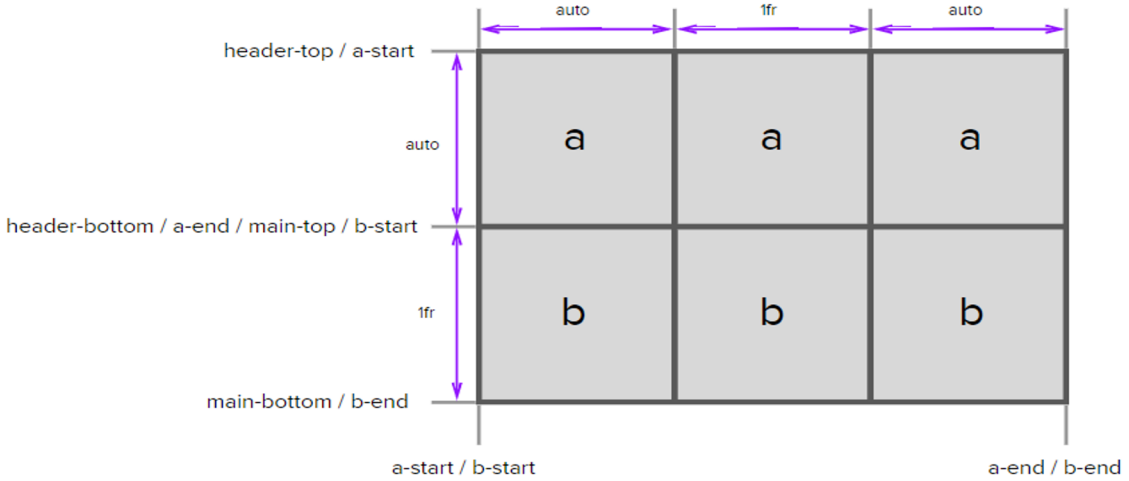
The grid-template-area Property

grid-template-areas: none |
<string>+

The grid-template Property

grid-template: none | [<‘grid-template-rows’> / <‘grid-template-columns’>] | [
<line-names>? <string> <track-size>? <line-names>?]+ [/ <explicit-track-list>]?

```
grid-template-areas: "a a a"
                     "b b b";
grid-template-rows: [header-top]
auto [header-bottom main-top]
1fr [main-bottom];
grid-template-columns: auto 1fr
auto;
```



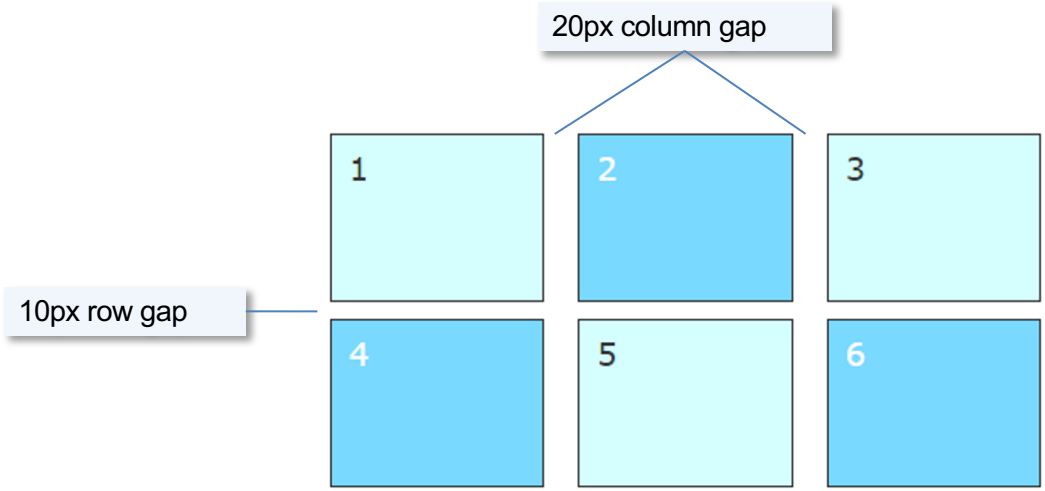
The grid-column-gap and grid-row-gap Properties

grid-column-gap: <length> |
<percentage>
grid-row-gap: <length> | <percentage>

The grid-gap Property

grid-gap: <‘grid-row-gap’> <‘grid-column-gap’>?

grid-gap: <‘grid-row-gap’> <‘grid-column-gap’>




```
div#outer {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
    grid-auto-rows: 100px;
    grid-gap: 10px 20px
}
```

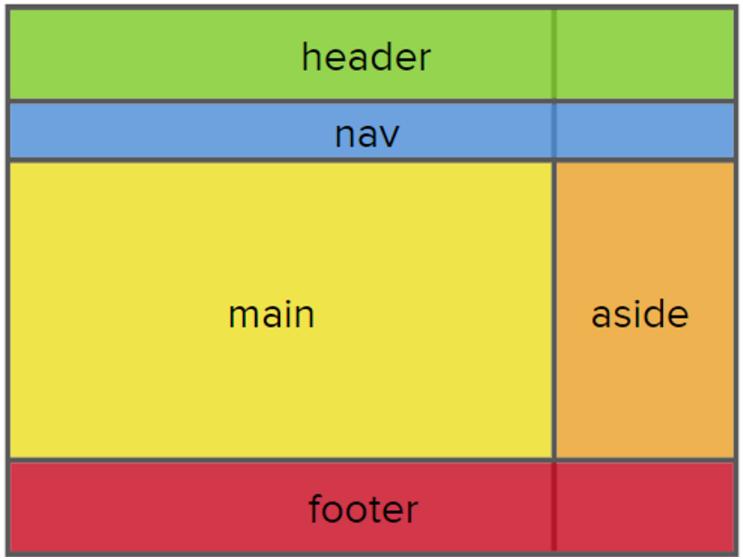
The grid Property

grid: <‘grid-template’> | <‘grid-template-rows’> / [auto-flow && dense?] <‘grid-auto-columns’>? | [auto-flow && dense?] <‘grid-auto-rows’>? / <‘grid-template-columns’>

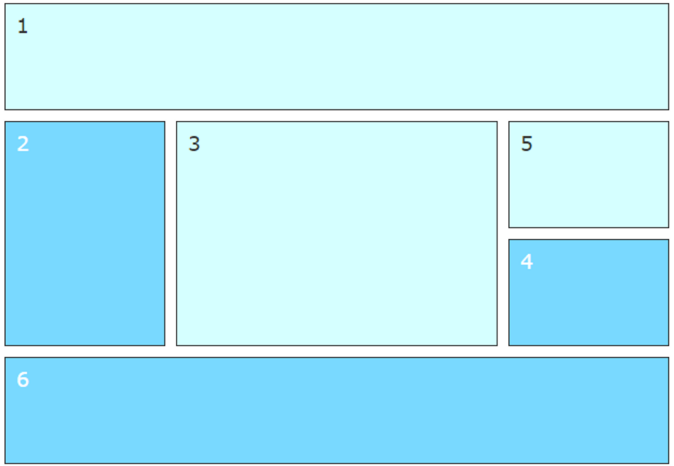
```
/* Initial values of each sub-property */
grid-template-rows: none
grid-template-columns: none
grid-template-areas: none
grid-auto-rows: auto
grid-auto-columns: auto
grid-auto-flow: row
grid-column-gap: 0
grid-row-gap: 0
```

Ordering Grid Items

```
<body>
  <header>...</header>
  <main>...</main>
  <nav>...</nav>
  <aside>...</aside>
  <footer>...</footer>
</body>
```



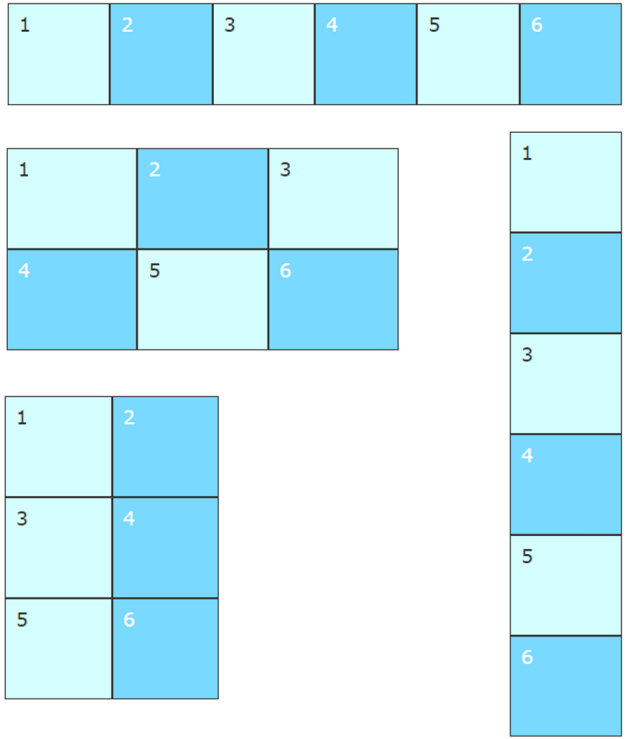

```
body {
  display: grid;
  grid: "header header"
        "nav nav"
        "content sidebar"
        "footer footer";
  grid-template-columns: 1fr 25%;
}

header {
grid-area: header;
 }

nav {
 grid-area: nav;
}

footer {
 grid-area: footer;
}
```




```
div#outer {
    display: grid;
    grid-template-columns: 150px minmax(300px, auto) 150px;
    grid-template-rows: repeat(4, 100px);
    grid-gap: 10px;
    grid-template-areas: "header header header"
                         "nav main side1"
                         "nav main side2"
                         "footer footer footer";
}

div#inner1 {
    grid-area: header;
}

div#inner2 {
    grid-area: nav;
}

div#inner3 {
    grid-area: main;
}

div#inner4 {
    grid-area: side2;
}

div#inner5 {
    grid-area: side1;
}

div#inner6 {
    grid-area: footer;
}
```

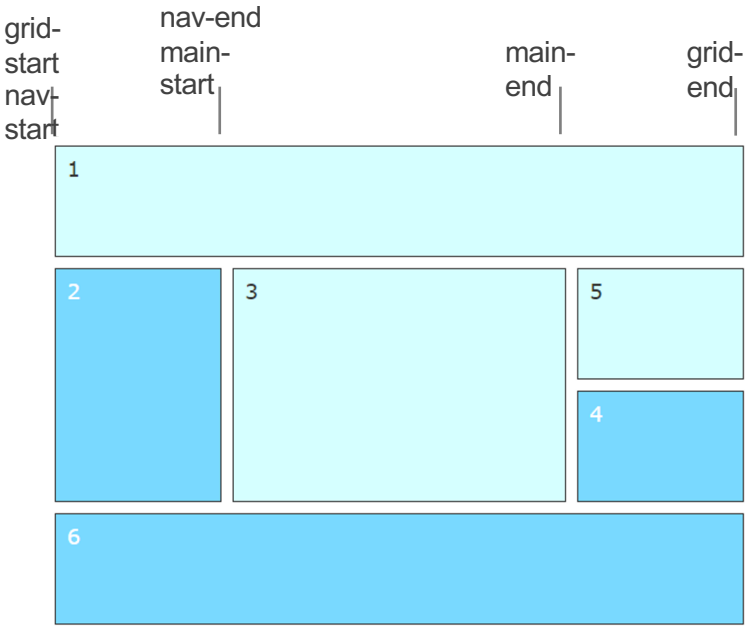
named grid areas defined within the grid

grid areas associated with grid items



automatically fill the column track with as many grid items as the space allows

```
div#outer {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
    grid-auto-rows: 100px;
}
```



grid line names

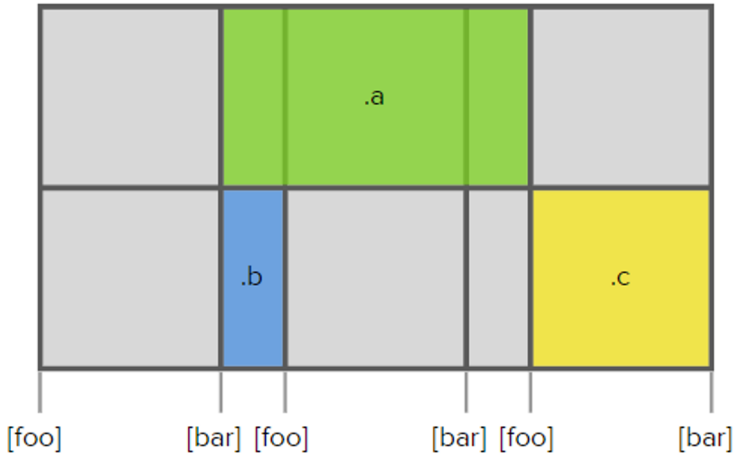
```
div#outer {
    display: grid;
    grid-template-columns: [grid-start nav-start] 150px
                           [nav-end main-start] minmax(300px, auto)
                           [main-end] 150px
                           [grid-end];
    grid-auto-rows: 100px;
    grid-gap: 10px;
}

div#inner1 {
    grid-column: grid-start/grid-end;
}

div#inner6 {
    grid-column: 1/-1;
}

div#inner2 {
    grid-column: nav-start/nav-end;
    grid-row: span 2;
}

div#inner3 {
    grid-column: main-start/main-end;
    grid-row: span 2;
}

div#inner5 {
    grid-column: 3;
    grid-row: 2;
}
```

grid line names used to define the grid areas

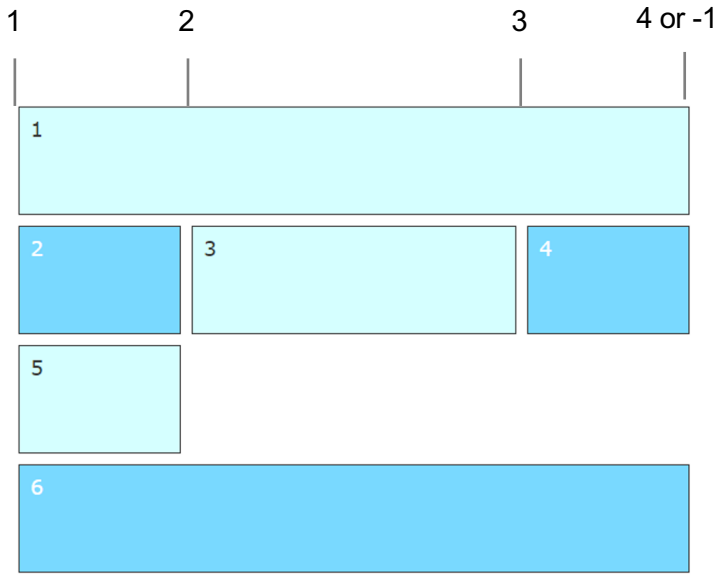
Placing Grid Items

The grid-column-start, grid-column-end and grid-row-start, grid-row-end Properties

grid-column-start: auto | <custom-ident> | [<integer> && <custom-ident>?] | [span && [<integer> || <custom-ident>]]
grid-column-end: auto | <custom-ident> | [<integer> && <custom-ident>?] | [span && [<integer> || <custom-ident>]]
grid-row-start: auto | <custom-ident> | [<integer> && <custom-ident>?] | [span && [<integer> || <custom-ident>]]
grid-row-end: auto | <custom-ident> | [<integer> && <custom-ident>?] | [span && [<integer> || <custom-ident>]]

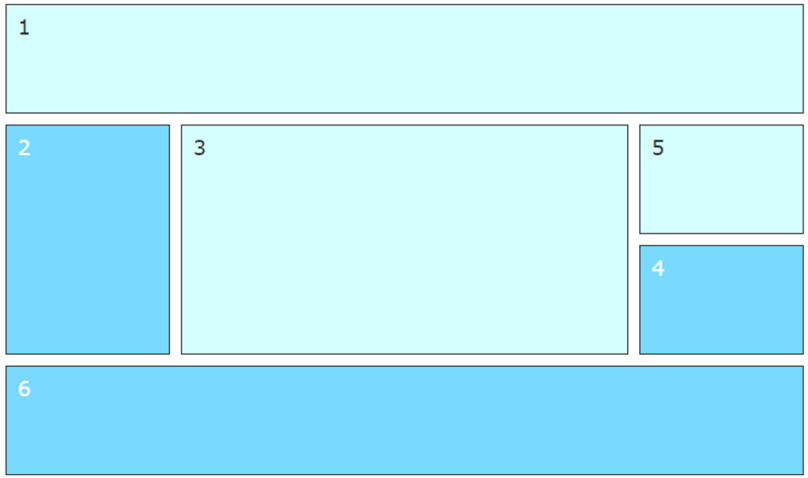



```
.a {
grid-column-start: 1 bar;
grid-column-end: 3 foo;
 }
.b {
grid-column-start: 1 bar;
}
.c {
 grid-column-start: -1 foo;
 }
```




```
div#outer {
    display: grid;
    grid-template-columns: 150px minmax(300px, auto) 150px;
    grid-auto-rows: 100px;
    grid-gap: 10px;
}

div#inner1 {
    grid-column-start: 1;
    grid-column-end: -1;
}

div#inner6 {
    grid-column: 1/-1;
}
```

grid item extends from grid line 1 to -1 (the last grid line)




```
div#outer {
    display: grid;
    grid-template-columns: 150px minmax(300px, auto) 150px;
    grid-auto-rows: 100px;
    grid-gap: 10px;
}

div#inner1 {
    grid-column-start: 1;
    grid-column-end: -1;
}

div#inner6 {
    grid-column: 1/-1;
}

div#inner2 {
    grid-row: span 2;
}

div#inner3 {
    grid-row: span 2;
}

div#inner5 {
    grid-column: 3;
    grid-row: 2;
}
```

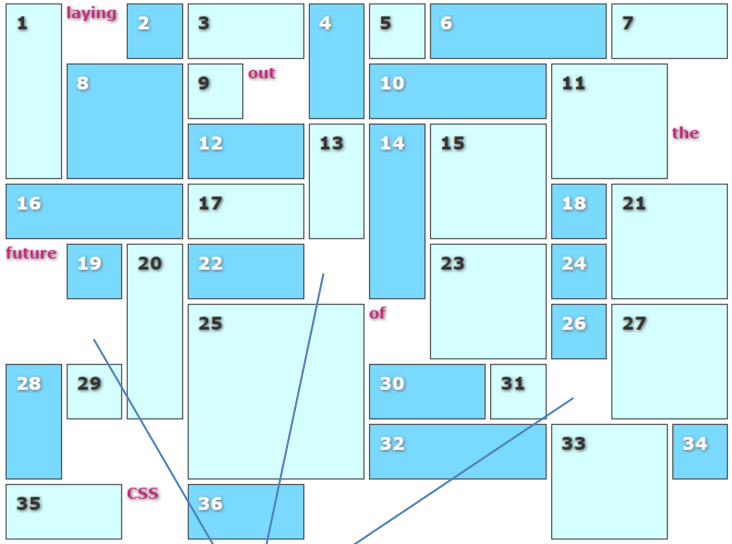
grid items span 2 row

grid item moved to column 3 and row 2

The Auto-placement Algorithm

1. Generate anonymous grid items
2. Place elements with explicit grid positions
3. Place elements with a explicit row positions but no column position
4. Determine the number of columns in the implicit grid
5. Position the remaining elements

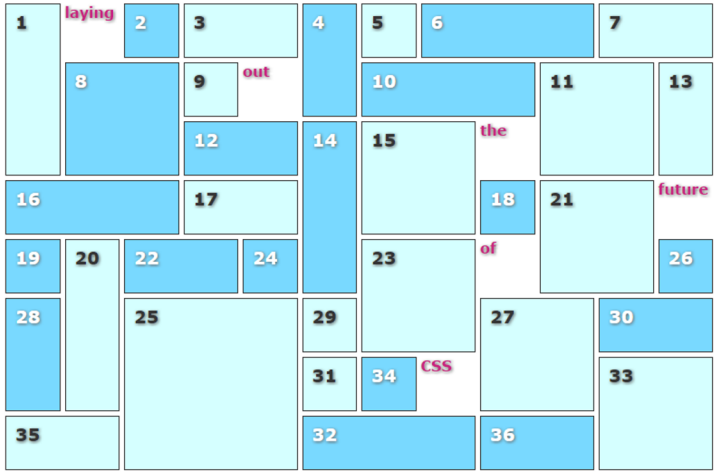
anonymous grid item



sparse placement causes empty spaces in the middle of the layout

```
div#outer {
   display: grid;
   grid-template-columns: repeat(auto-fill, minmax(60px, 1fr));
   grid-auto-rows: minmax(60px, auto);
   grid-gap: 5px;
}

div.w1h2 {
    grid-row: span 2;
}

div.w2h1 {
    grid-column: span 2;
}

div.w2h2 {
    grid-column: span 2;
    grid-row: span 2;
}

div.w1h3 {
    grid-row: span 3;
}

div.w3h1 {
    grid-column: span 3;
}

div.w3h3 {
    grid-row: span 3;
    grid-column: span 3;
}
```

grid items spanning multiple columns and/or rows

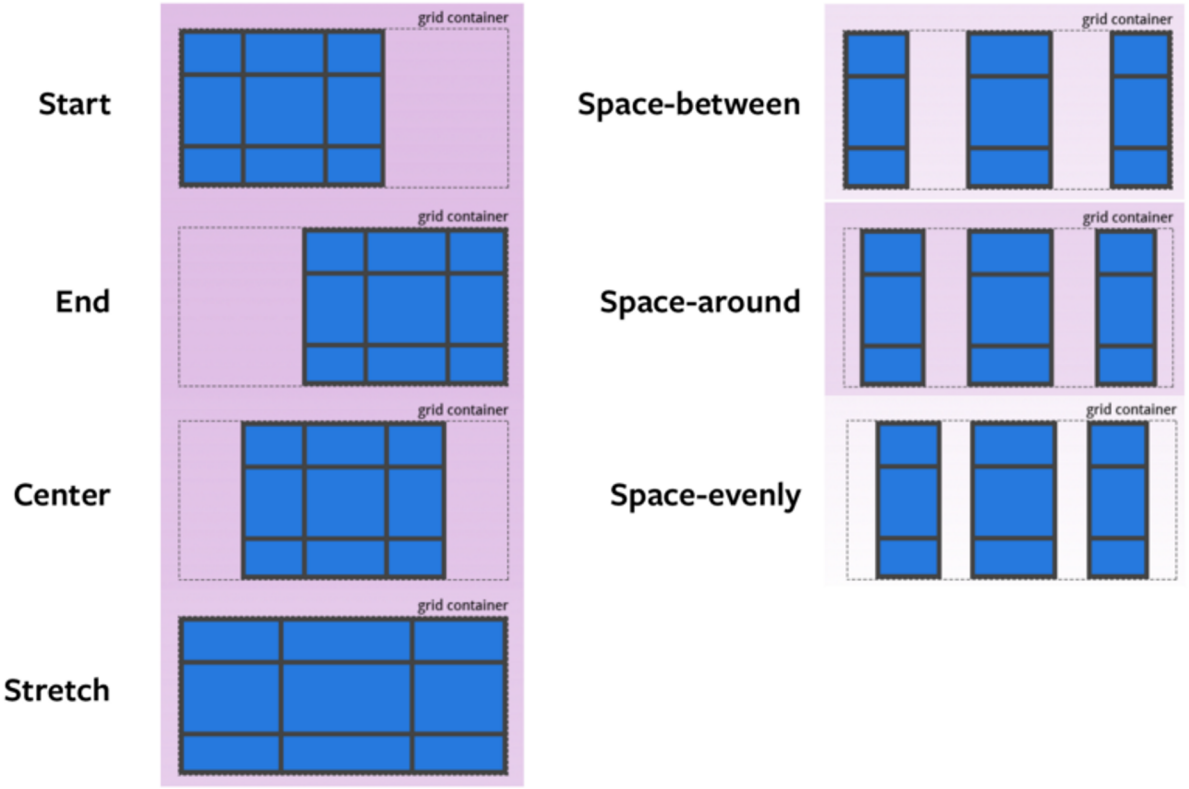



```
div#outer {
   display: grid;
   grid-template-columns: repeat(auto-fill, minmax(60px, 1fr));
   grid-auto-rows: minmax(60px, auto);
   grid-gap: 5px;
   grid-auto-flow: dense;
}


div.w1h2 {
    grid-row: span 2;
}

div.w2h1 {
    grid-column: span 2;
}

div.w2h2 {
    grid-column: span 2;
    grid-row: span 2;
}

div.w1h3 {
    grid-row: span 3;
}

div.w3h1 {
    grid-column: span 3;
}

div.w3h3 {
    grid-row: span 3;
    grid-column: span 3;
}
```

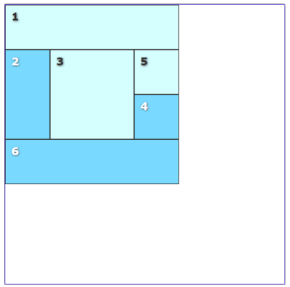
setting the grid-auto-flow to dense removes sparse cells at the expense of document order

Grid Alignment: Aligning the Grid and its Grid Items

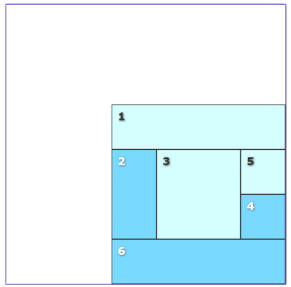
justify-content: center | start | end | space-between | space-around | space-evenly



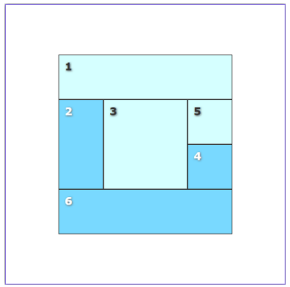
Justifying and Aligning



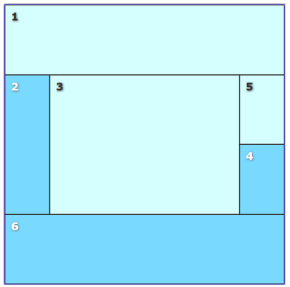
justify-content:
start;
align-content: start;



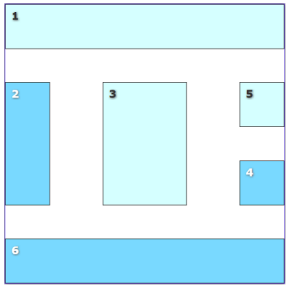
justify-content:
end;
align-content: end;



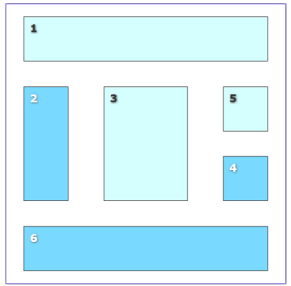
justify-content:
center;
align-content: center;



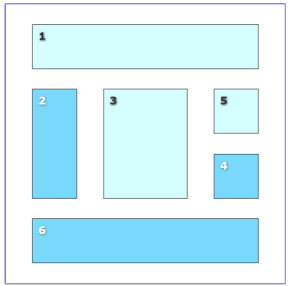
justify-content:
stretch;
align-content: stretch;



justify-content:
between;
align-content: between;



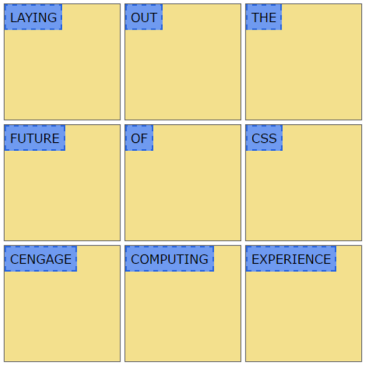
justify-content:
around;
align-content: around;



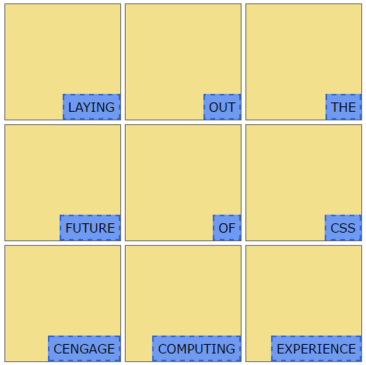
justify-content:
evenly;
align-content:
evenly;

justify-content: (*column axis*)
align-content: (*row axis*)

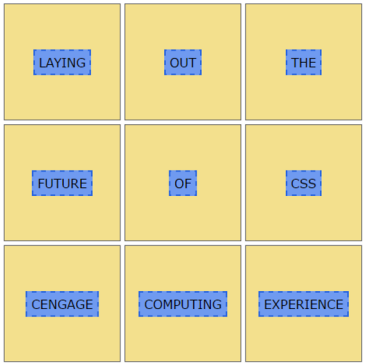
start	aligns the grid to the top or left end of the grid container
end	aligns the grid to the bottom or right end of the grid container
center	aligns the grid in the center of the grid container
stretch	resizes the grid items to allow the grid to fill the full width or height of the grid container (*the default*)
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



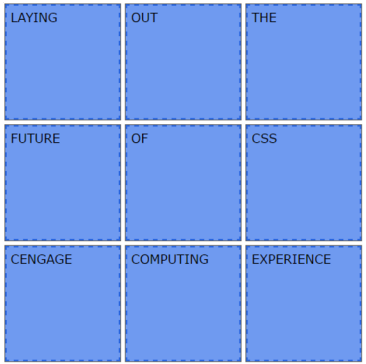
justify-items:
start;
align-items: start;



justify-items:
end;
align-items: end;



justify-items:
center;
align-items: center;

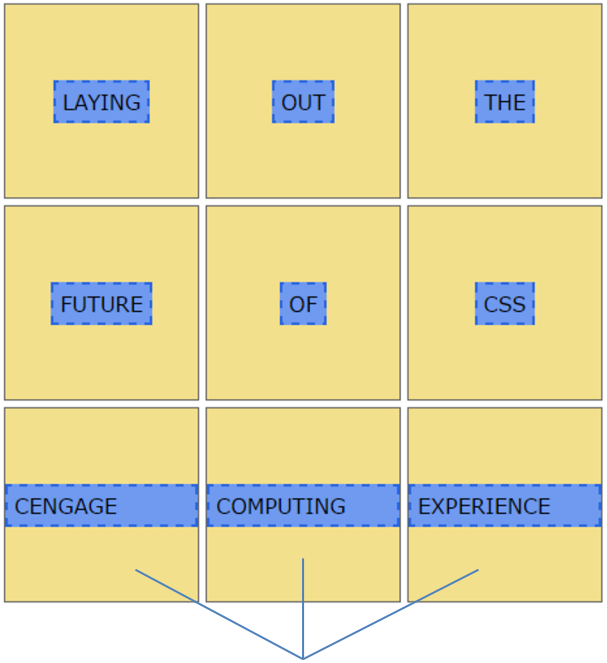


justify-items:
stretch;
align-items: stretch;

justify-items: (*column axis*)
align-items: (*row axis*)

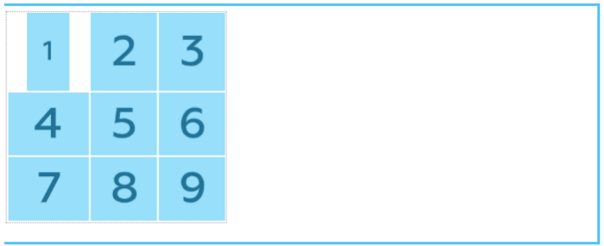
start	aligns the content to the top or left of the grid item area
end	aligns the content to the bottom or right of the grid item area
center	aligns the content to the center (horizontally or vertically) of the grid item area
stretch	stretches the content to cover the whole grid area (*the default*)

justify-self: (*column axis*)
align-self: (*row axis*)



justify-self:
stretch;
align-self: center;

start	aligns the content to the top or left of the grid item area
end	aligns the content to the bottom or right of the grid item area
center	aligns the content to the center (horizontally or vertically) of the grid item area
stretch	stretches the content to cover the whole grid area (*the default*)




```
.parent {
  display: grid;
}
.child {
  justify-self: center;
}
```


header
sidebar
Content-1
Content-2
Content-3
footer

FE Online UA Training Course Feedback

I hope that you will find this material useful.

If you find errors or inaccuracies in this material or know how to improve it, please report on to the electronic address:

serhii_shcherbak@epam.com

With the note [FE Online UA Training Course Feedback]

Thank you.

Q&A

UA Frontend Online LAB