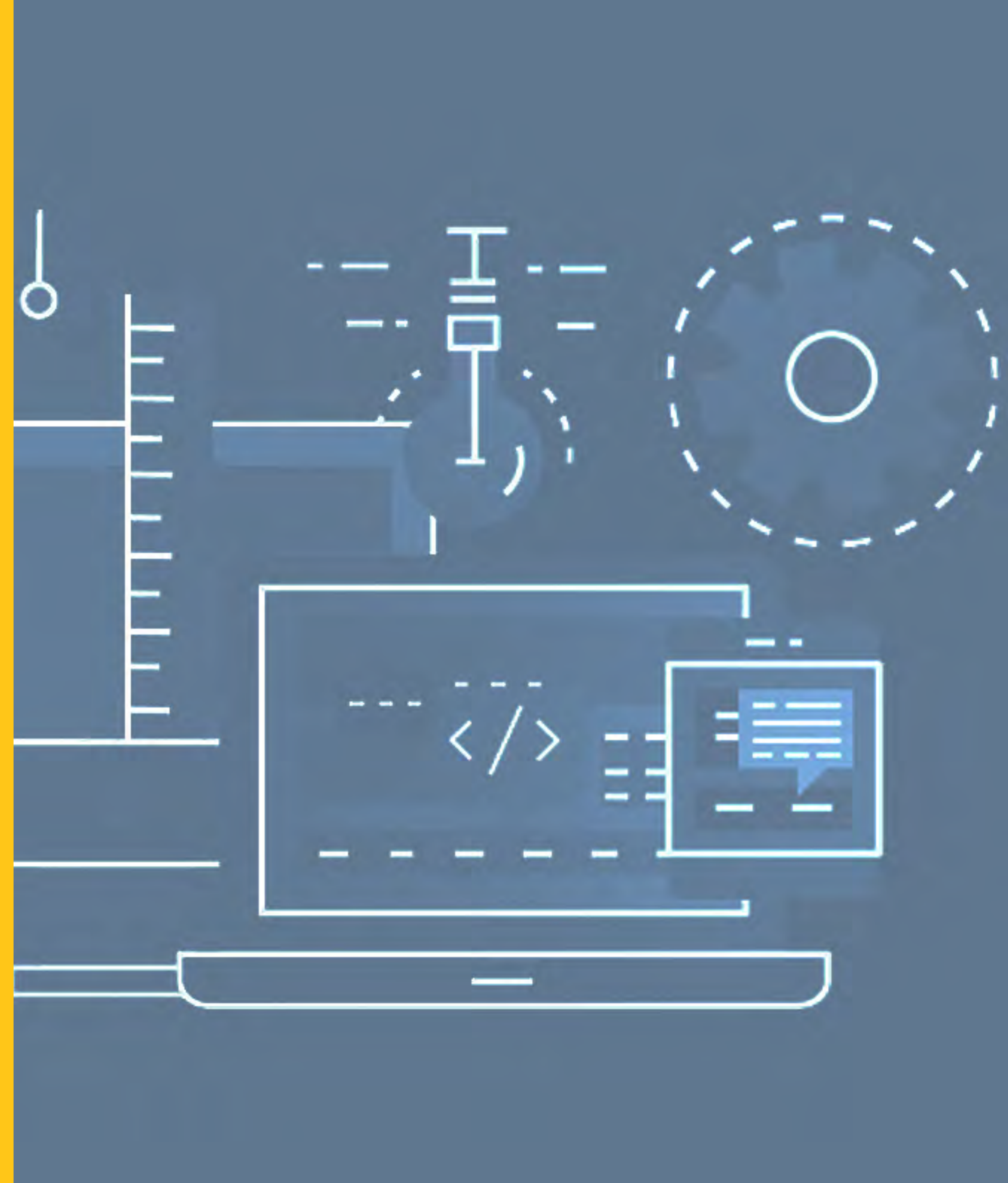


SHELLY GRAHAM, 02/11/2020

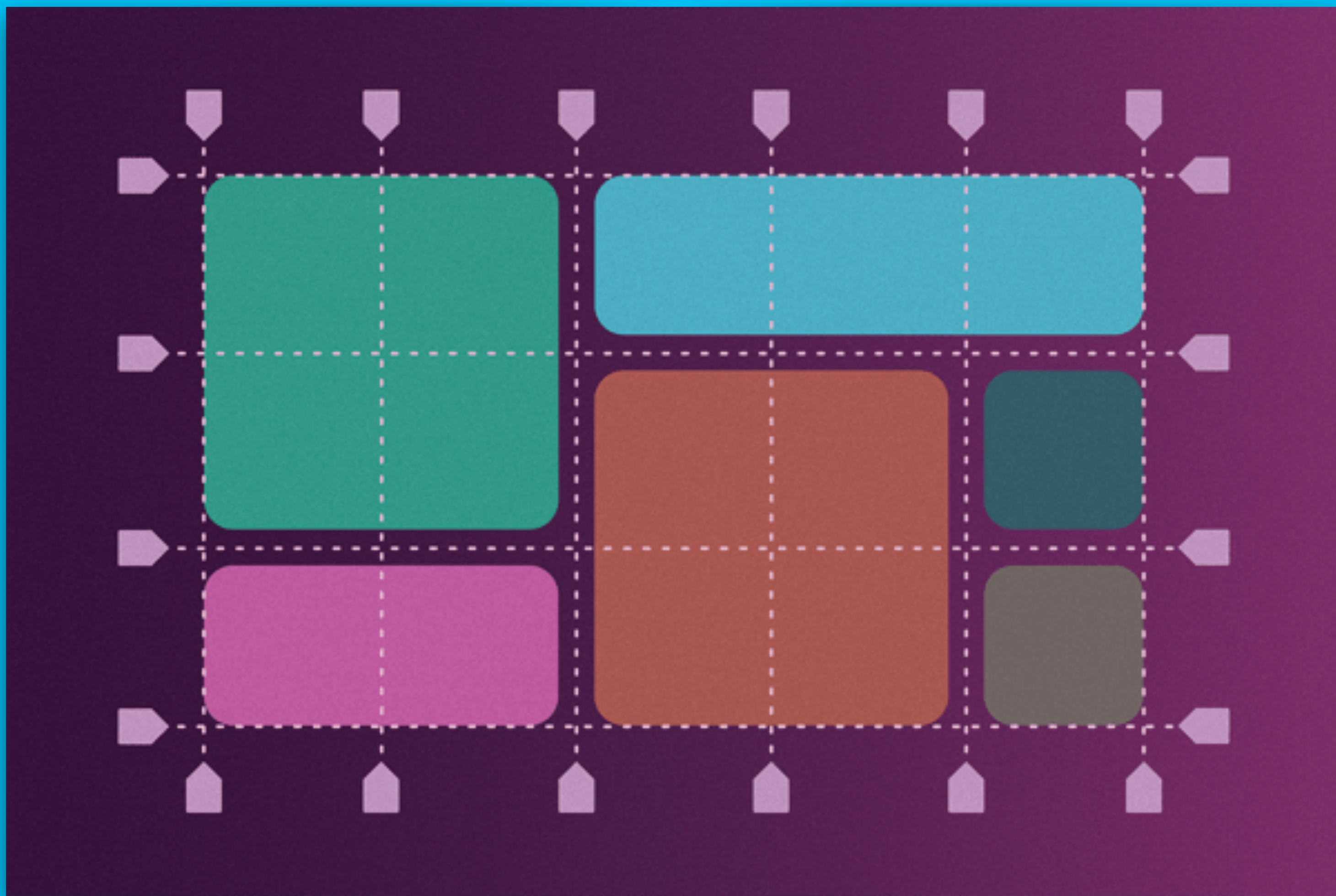
WEB DEV 3

WINTER 2021

Week 6: CSS Grid



WEEK 6: CSS GRID



RECAP: ROW OR COLUMN - YOU CAN'T HAVE BOTH

- Flexbox lets you layout either in row OR column form.
- Therefore you create HTML elements aka Flex Containers, simply to flex them in a certain direction - not efficient!
- You can't define a specific row and column amount but are rather forced to make the content fit in either row or column formats

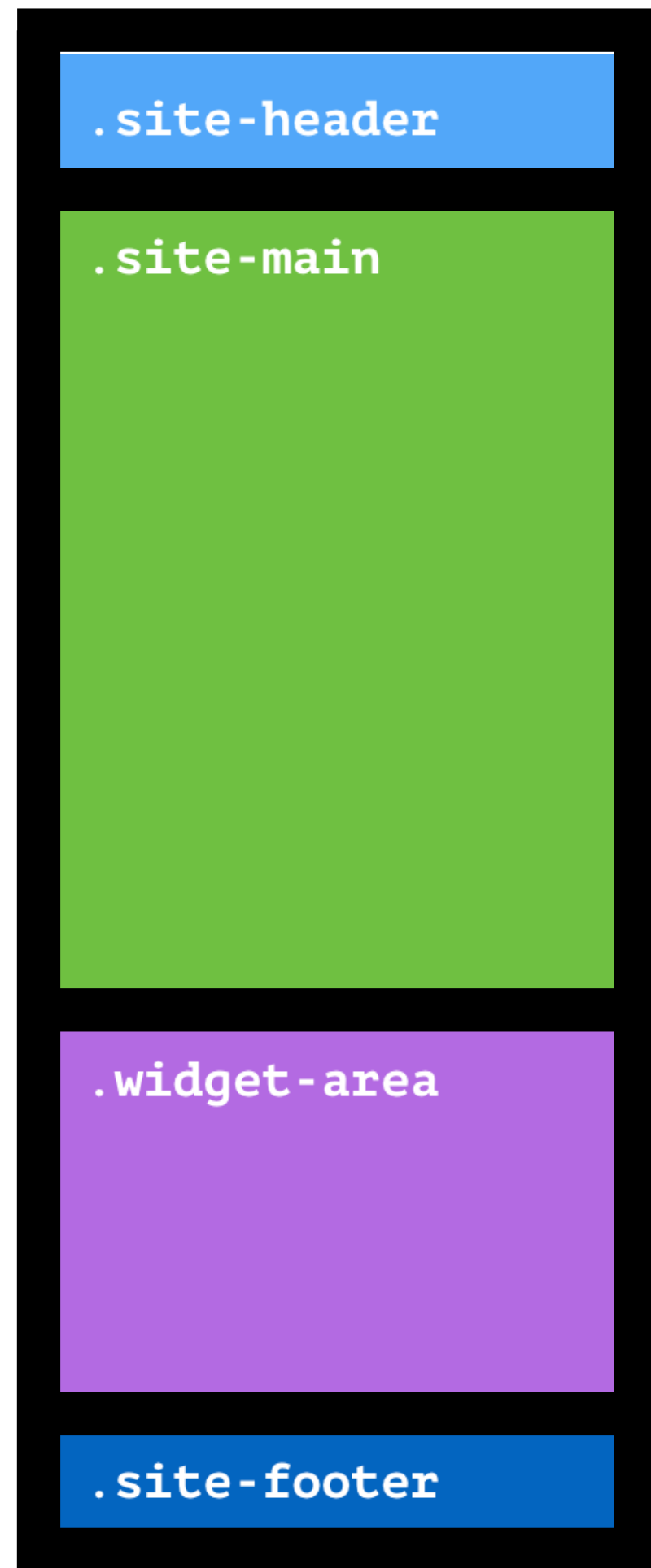


WHAT IS A GRID?

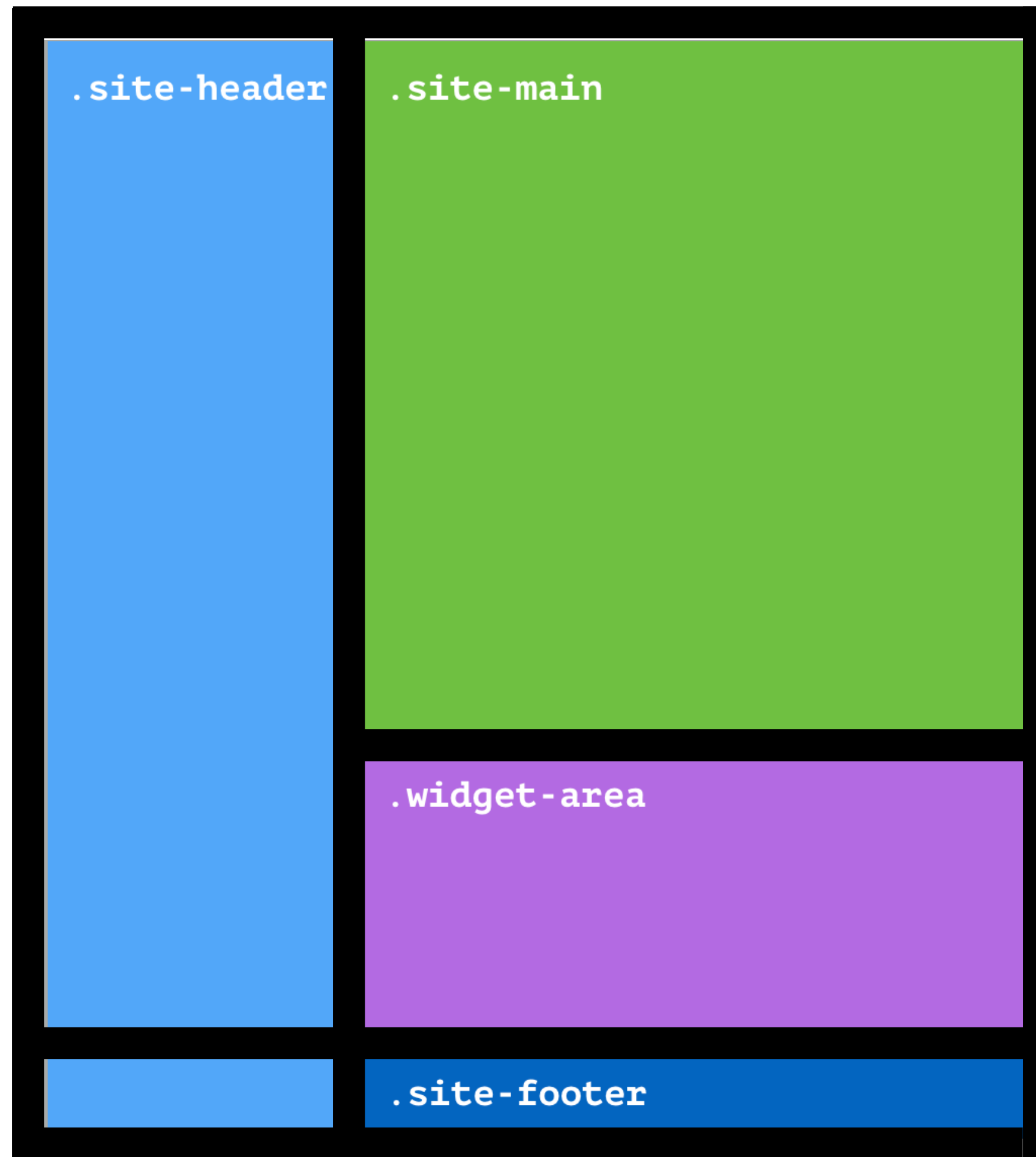
- A network of lines that cross each other to form a series of squares or rectangles
- No, but really?!



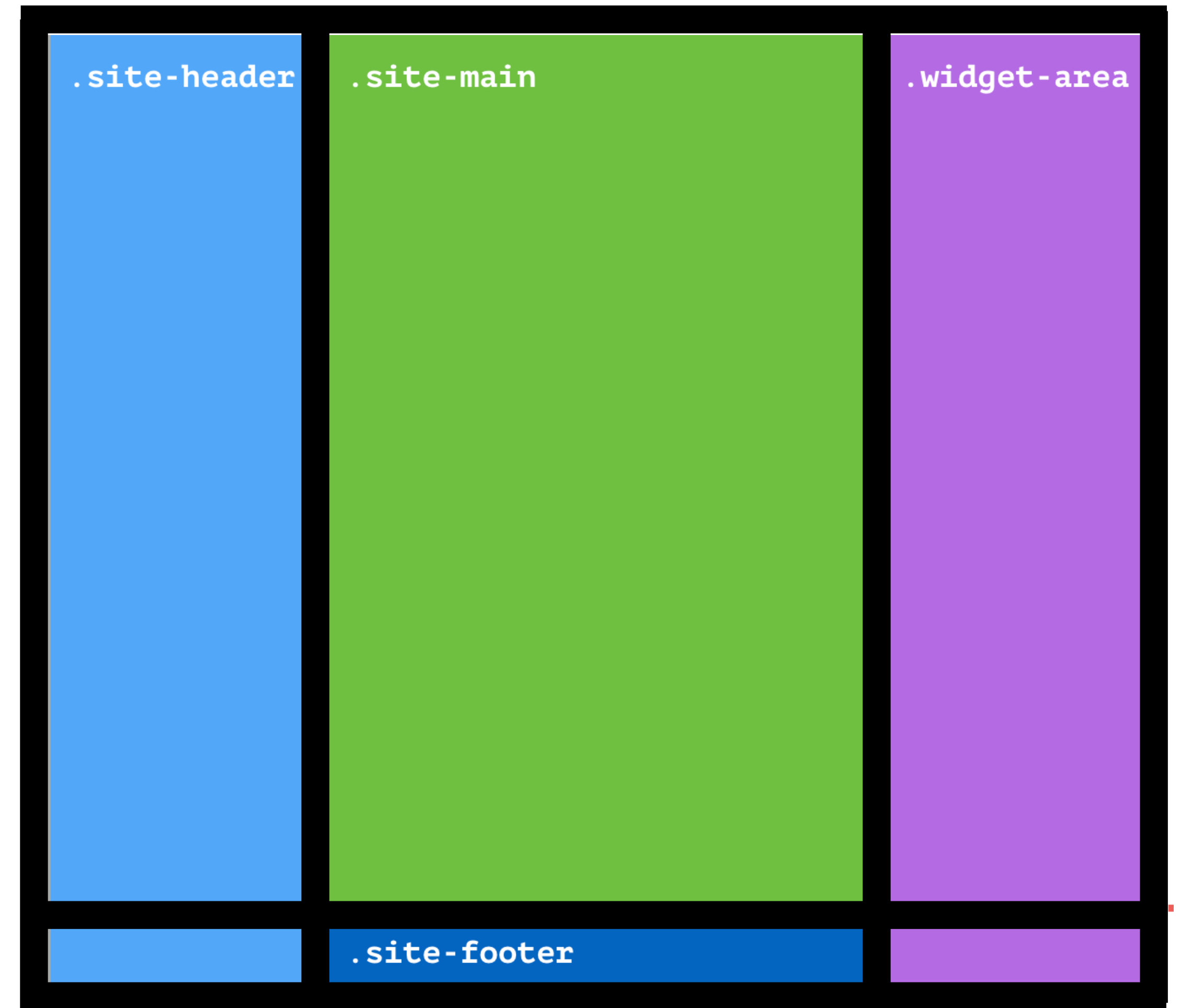
1. MOBILE



2. TABLET



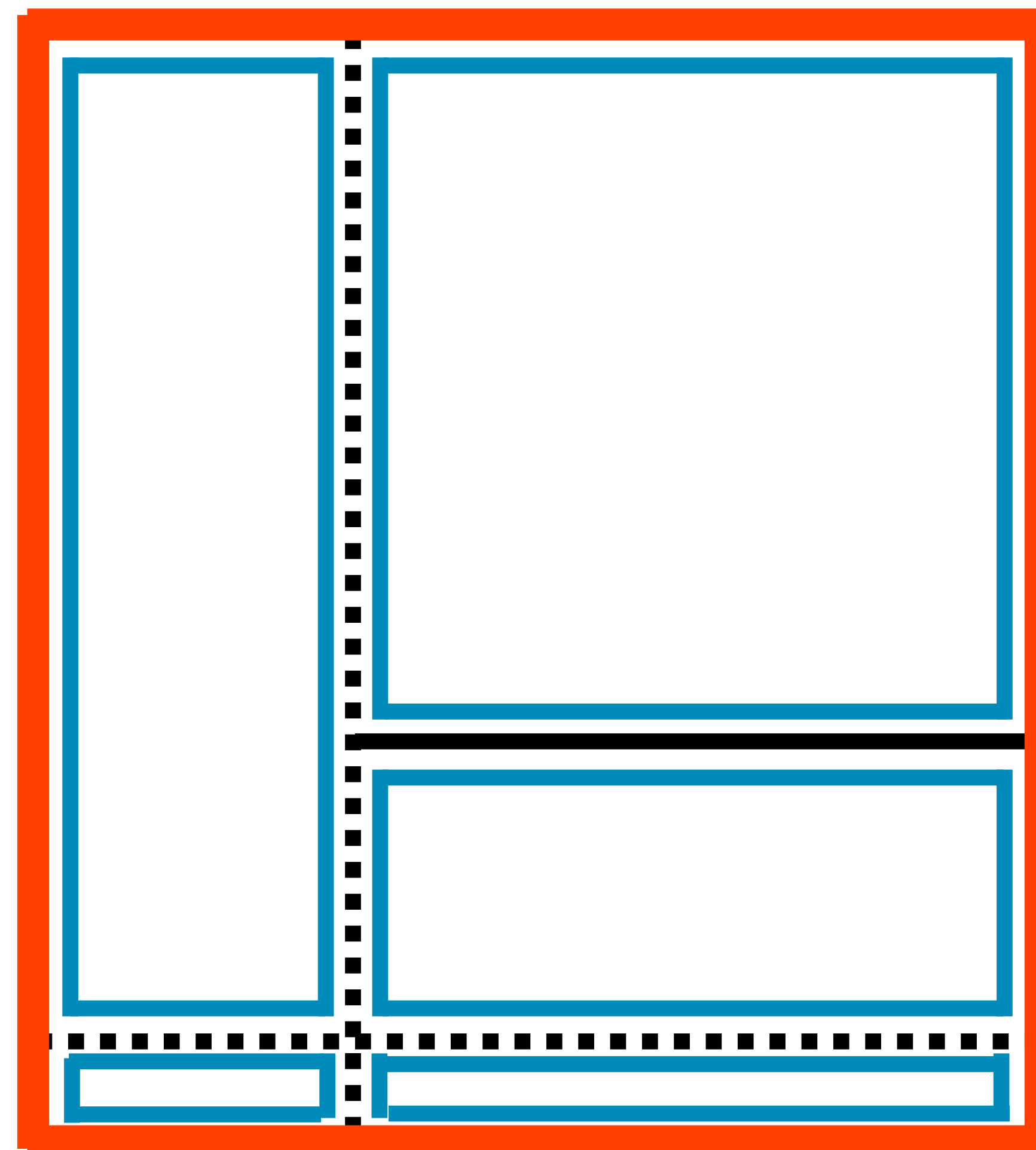
3. DESKTOP





IT ALL STARTS WITH...



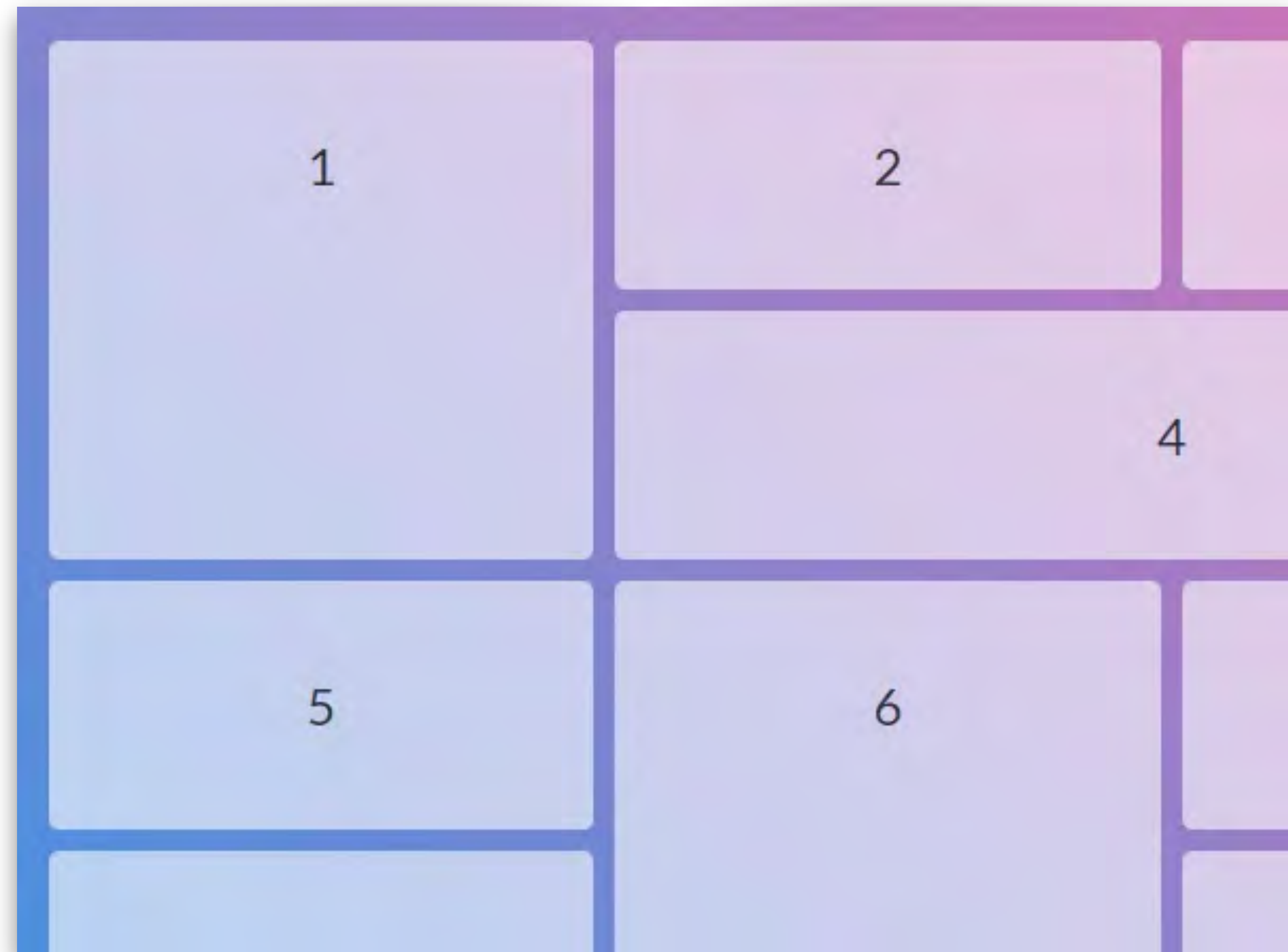


Grid Container

Grid Items

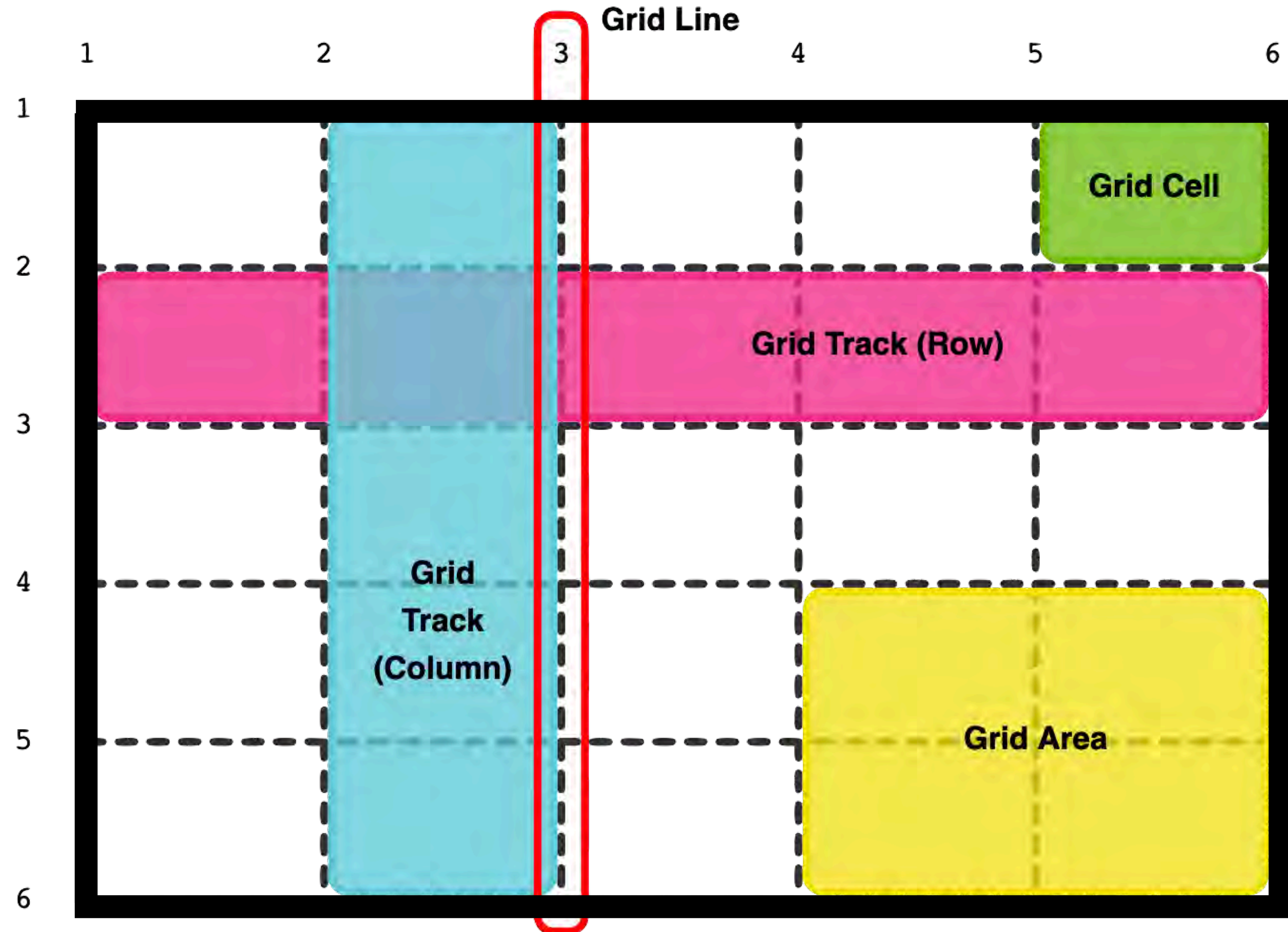
STRUCTURE OF CSS GRID

- Consists of rows and columns
- Rows and columns are defined by lines
- Each block is a grid cell, like in Excel
- You can merge cells, add cells, add rows, columns - also just like in Excel!
- Grid items can also be grid containers



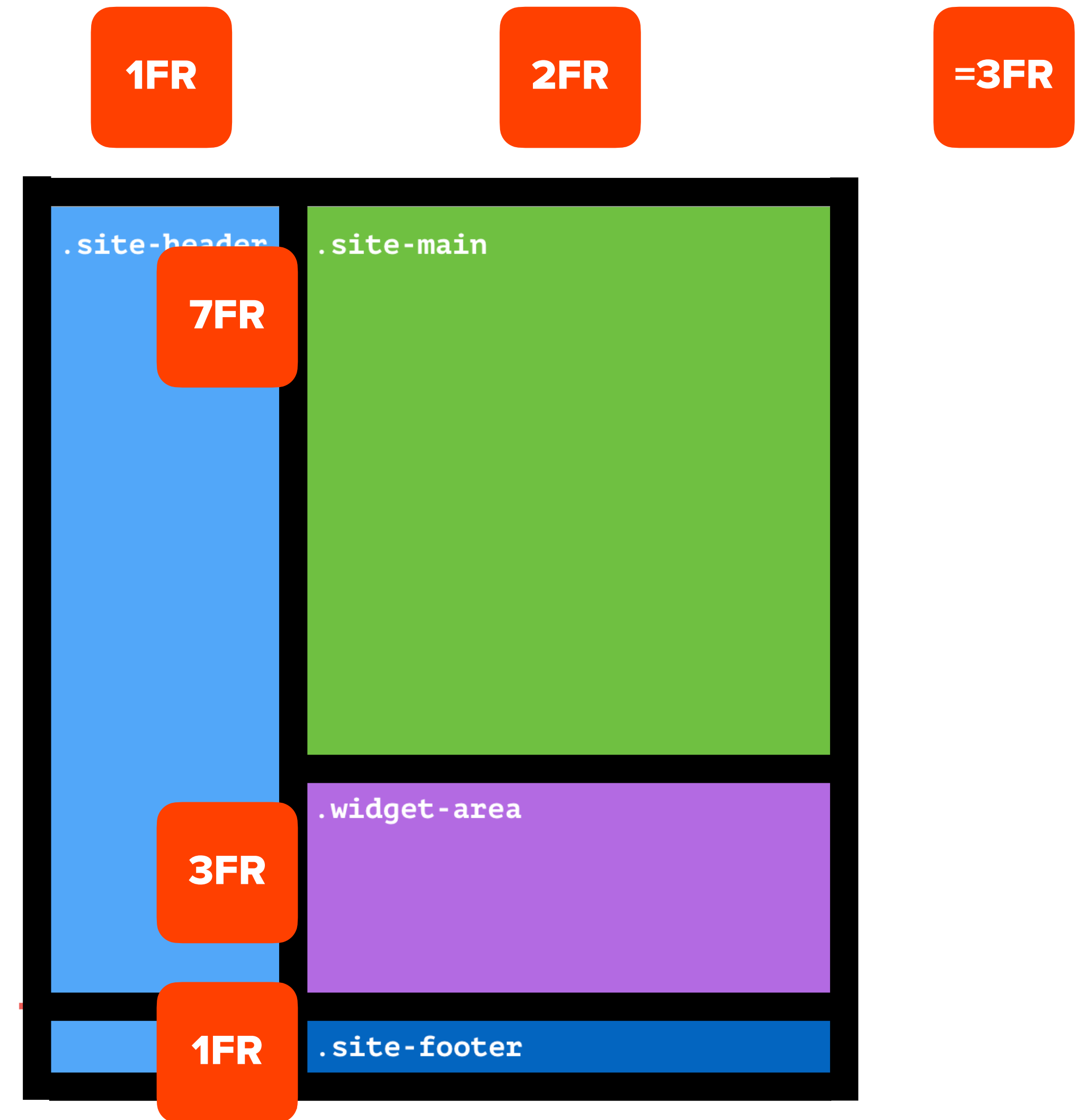
STRUCTURE OF CSS GRID

- **Grid Cell:** One block of content
- **Grid Line:** Divides grid into individual grid cells
 - **Explicit grid line:** manually set structure of grid
 - **Implicit grid line:** lines to accommodate content that has not been explicitly defined
- **Track:** One entire row or column
- **Area:** Square or rectangle of multiple cells



SPECIAL UNIT: FR

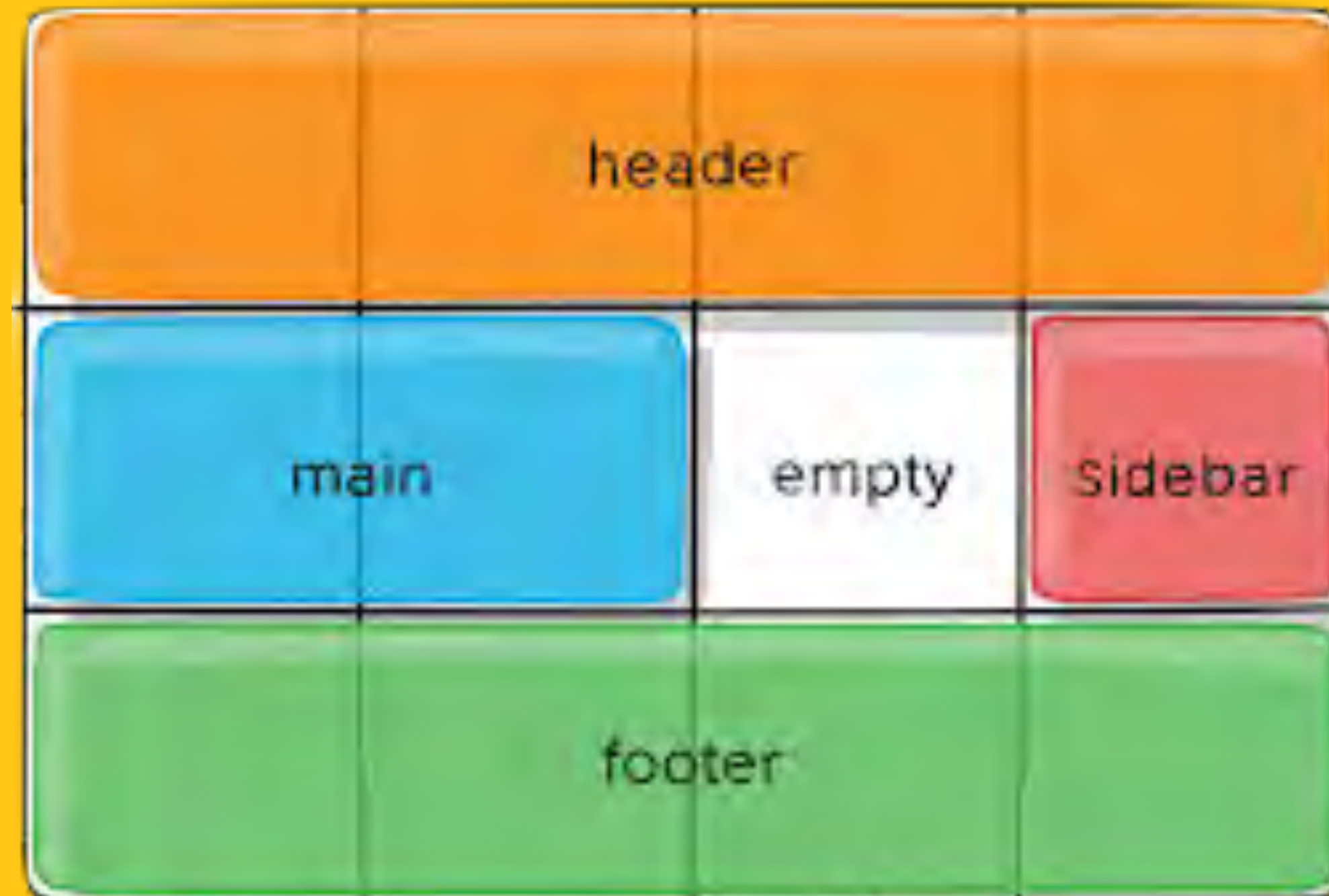
- fr = fraction aka **f**ree space
- Uses the remaining space of the grid
- Often best unit to use, even over percentages or vw/vh



CSS GRID IN A NUTSHELL:

1. DEFINE GRID
2. PLACE ITEMS IN GRID
3. WORLD PEACE

GRID BASICS



THE PARENT - TEMPLATE

DISPLAY: GRID

- Defines element as grid container
- Values:
 - `grid`: generates a block-level grid
 - `inline-grid`: generates an inline-level grid

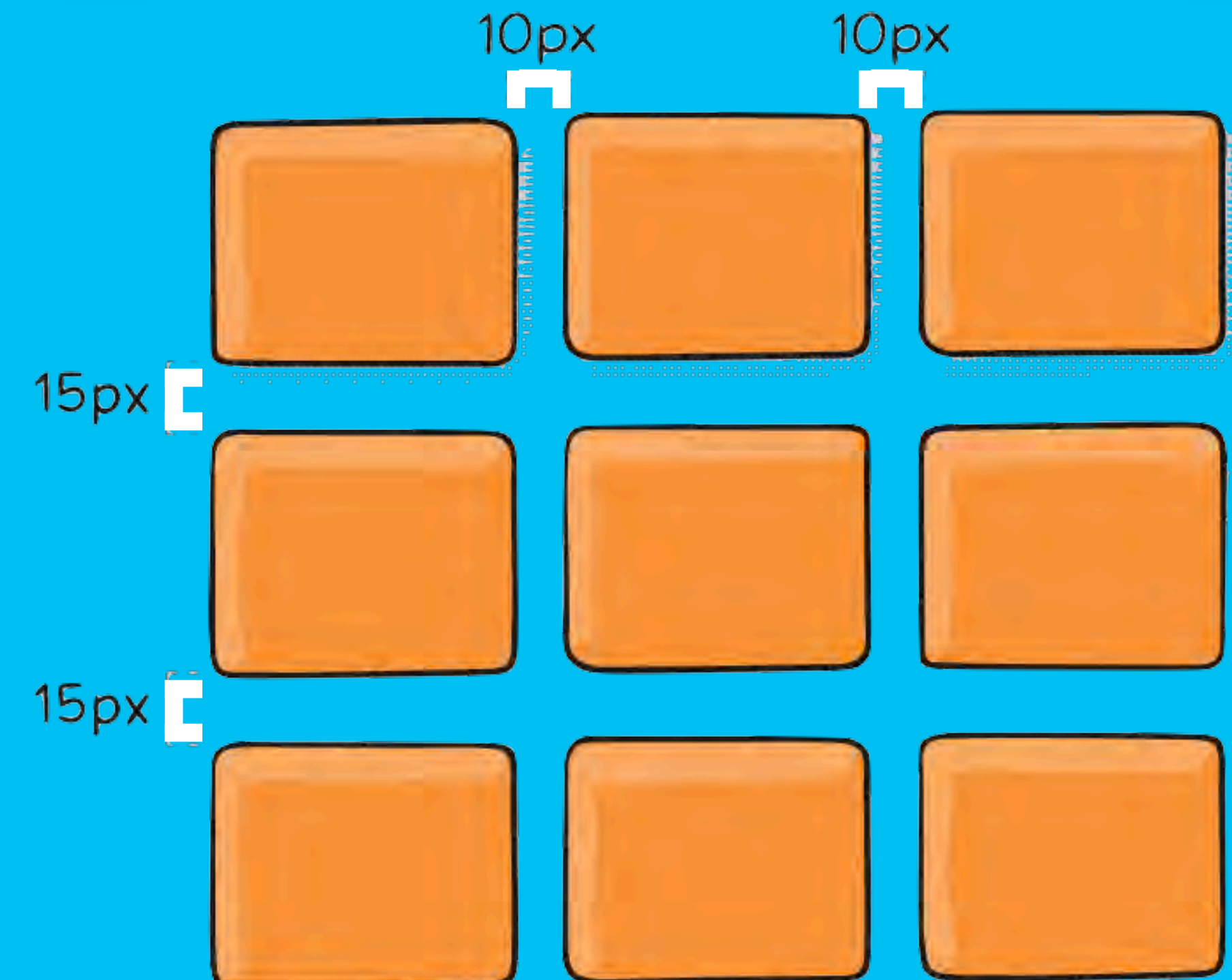
```
.grid-container {  
  display: grid  
  inline-grid;  
}
```

THE PARENT - TEMPLATE

GAPS

- row-gap
- column-gap
- grid-row-gap
- grid-column-gap
- **SHORTHAND:**
 - gap: grid-row-gap grid-column-gap
- Value:
 - Any unit (px, %, vw, rems...)

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: 80px auto 80px;  
  column-gap: 10px;  
  row-gap: 15px;  
  
  // OR  
  
  gap: 10px 15px;  
}
```



THE PARENT - TEMPLATE

GRID-TEMPLATE-COLUMNS GRID-TEMPLATE-ROWS

- Define # and width of rows:
 - grid-template-rows: [value]
- Define # and width of columns:
 - grid-template-columns: [value]
- Value can be:
 - Any unit (px, %, vw, rems...)
 - Any name you choose

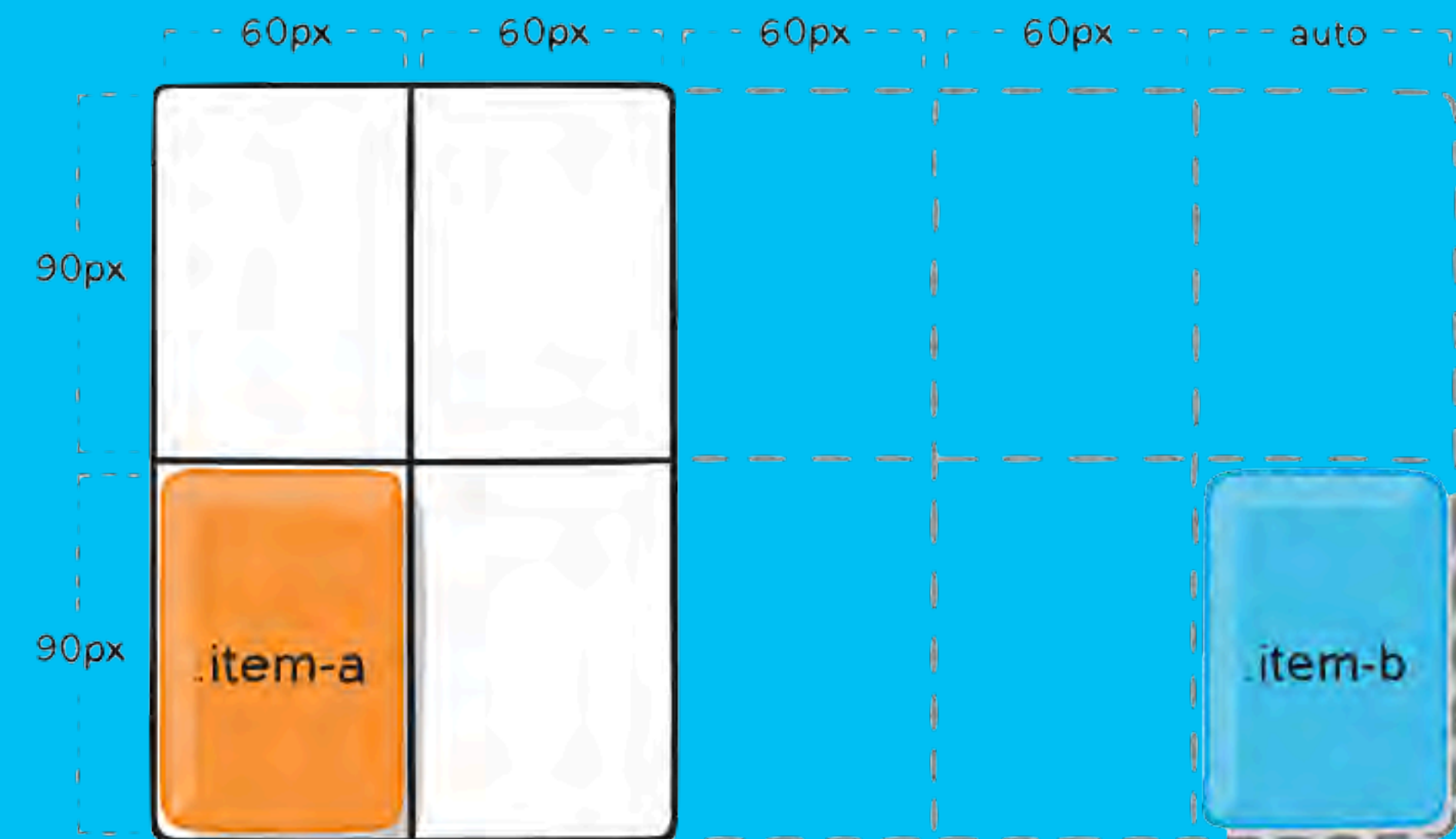
```
.grid-container {  
  display: grid;  
  grid-template-columns: 40px 50px auto 50px 40px;  
  grid-template-rows: 25% 100px auto;  
}
```

THE PARENT - TEMPLATE

GRID-AUTO-COLUMNS GRID-AUTO-ROWS

- Specifies the size of any auto-generated grid tracks = *implicit tracks*
- Implicit tracks get created when there are more grid items than cells in the grid or when a grid item is placed outside of the explicit grid

```
.grid-container {  
  display: grid;  
  grid-auto-columns: 60px;  
  grid-auto-rows: 200px;  
}
```



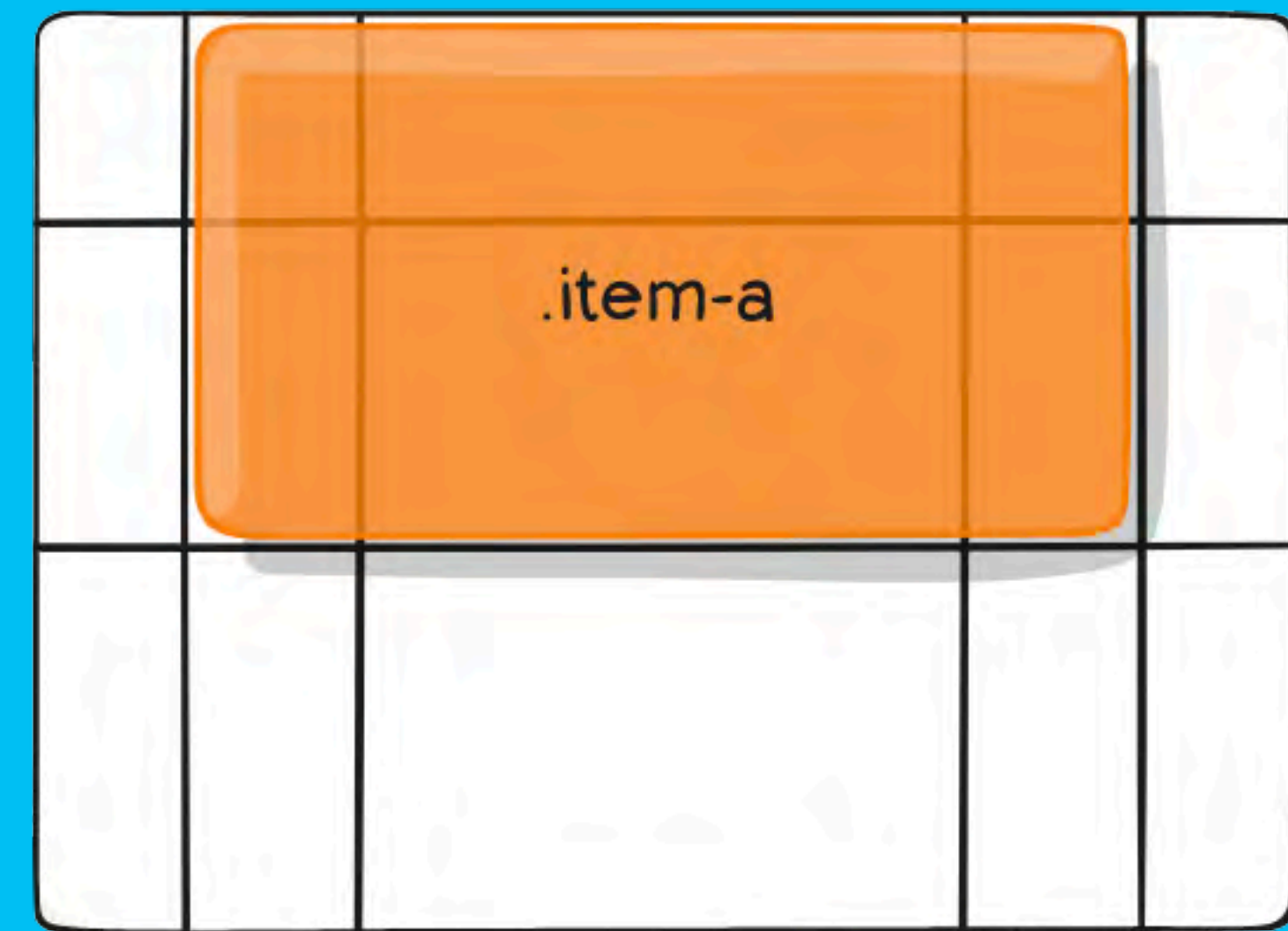
LETS CODE!

THE CHILDREN

DEFINE ROWS AND COLUMNS FOR CHILDREN

- Determines a grid item's location within the grid
- `grid-row-start`
- `grid-row-end`
- `grid-column-start`
- `grid-column-end`
- If no "grid-row/column end" is declared, the item will span 1 track by default

```
.grid-container {  
  display: grid;  
  grid-template-columns: 10% 10% 60% 10% 10%;  
  
  .grid-item-one {  
    grid-column-start: 2;  
    grid-column-end: 5;  
    grid-row-start: 1;  
    grid-row-end: 3;  
  }  
}
```



THE CHILDREN

DEFINE ROWS AND COLUMNS FOR CHILDREN

— SHORTCUTS:

- grid-row
- grid-column
- grid-area
 - Order: grid-row-start/
grid-column-start/
grid-row-end/
grid-column-end
 - OR: Simply give it a name!

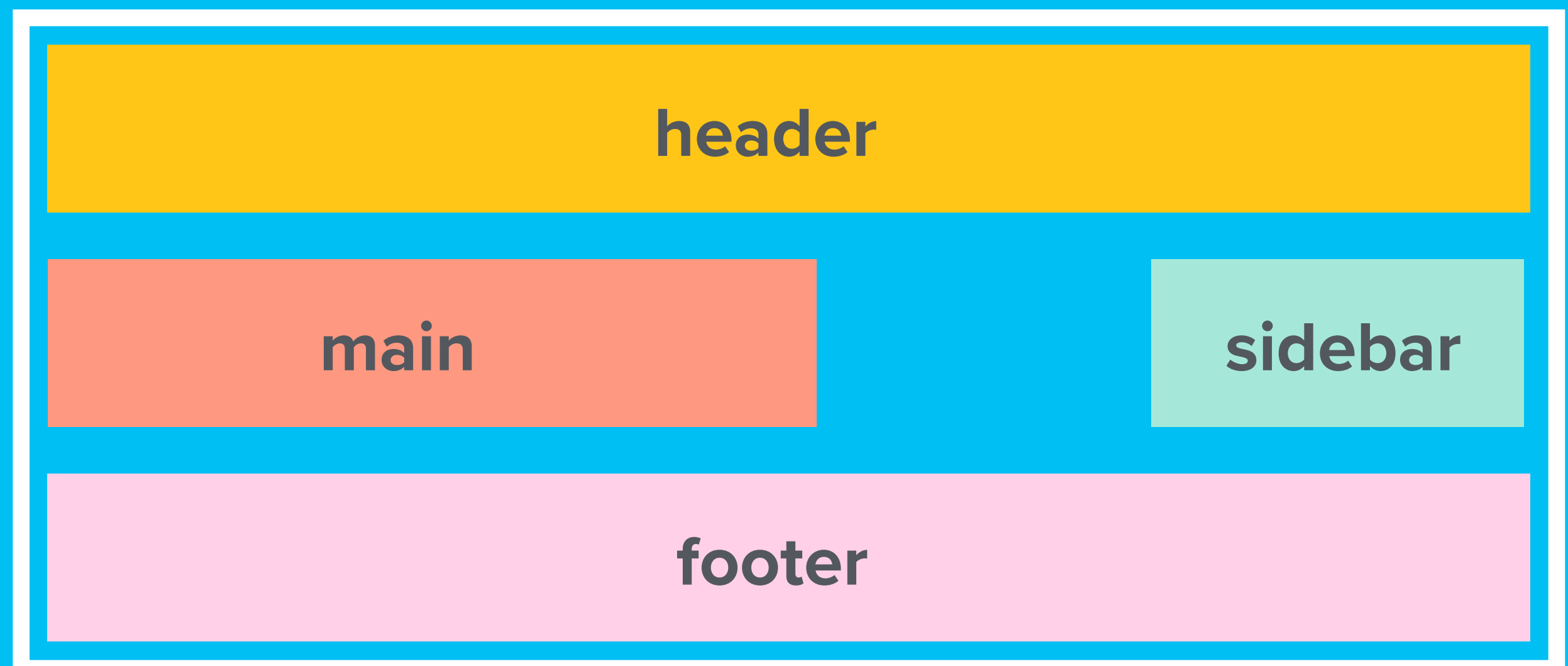
```
.grid-container {  
  display: grid;  
  grid-template-columns: 10% 10% 60% 10% 10%;  
  
  .grid-item-one {  
    grid-column-start: 2;  
    grid-column-end: 5;  
    grid-row-start: 1;  
    grid-row-end: 3;  
  
    ///// OR /////  
  
    grid-column: 2 / 5;  
    grid-row: 1 / 3;  
  
    ///// OR /////  
  
    grid-area: 1 / 2 / 3 / 5;  
  
    ///// OR /////  
  
    grid-area: main;  
  }  
}
```

THE PARENT - TEMPLATE

GRID-TEMPLATE-AREAS

- BIG ONE FOR MY VISUAL FOLKS!
- Defines a grid template by referencing the names of grid areas
- Easiest to work in lines
- Each line uses quotation marks
- If you want three rows, write grid-area 3 times, if you want four, write it out 4 times etc...

```
.grid-container {  
  display: grid;  
  grid-template-columns: 50px 50px 50px 50px;  
  grid-template-rows: auto;  
  grid-template-areas:  
    "header header header header"  
    "main   main   .   sidebar"  
    "footer footer footer footer";  
}
```



THE PARENT - TEMPLATE

GRID-TEMPLATE

— SHORTHAND:

- grid-template-rows
grid-template-columns
grid-template-areas
- Can't use repeat() function as the tracks are intended to visually line up one-to-one with the rows/columns

```
.grid-container {  
  display: grid;  
  // grid-template-columns: 10% 10% 60% 10% 10%;  
  // grid-template-rows: repeat(4, 1fr);  
  
  grid-template:  
    "header      header header header header" 1fr  
    "sidebar-left main  main  main  sidebar" 1fr  
    "sidebar-left main  main  main  ." 1fr  
    "footer      footer footer footer footer" 1fr  
  /  
  10% 10% 60% 10% 10%;  
}
```

- Write rows on end of each line
- Write columns after area names and “/”
- No commas at and of each line
- Make use of whitespace, it will help!

THE PARENT - TEMPLATE

JUSTIFY-ITEMS

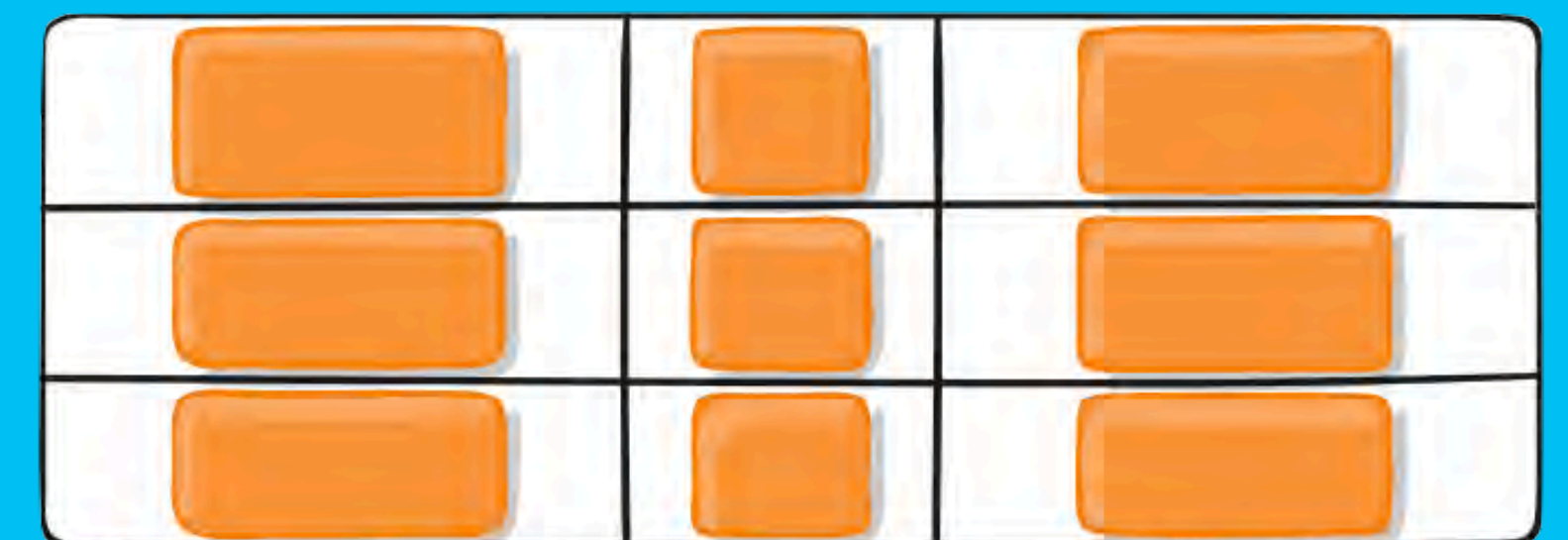
- Aligns grid items along the row axis = horizontal positioning
- Values:
 - start
 - end
 - center
 - stretch (default)

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  justify-items: stretch  
               start  
               end  
               center;  
}
```

stretch



center



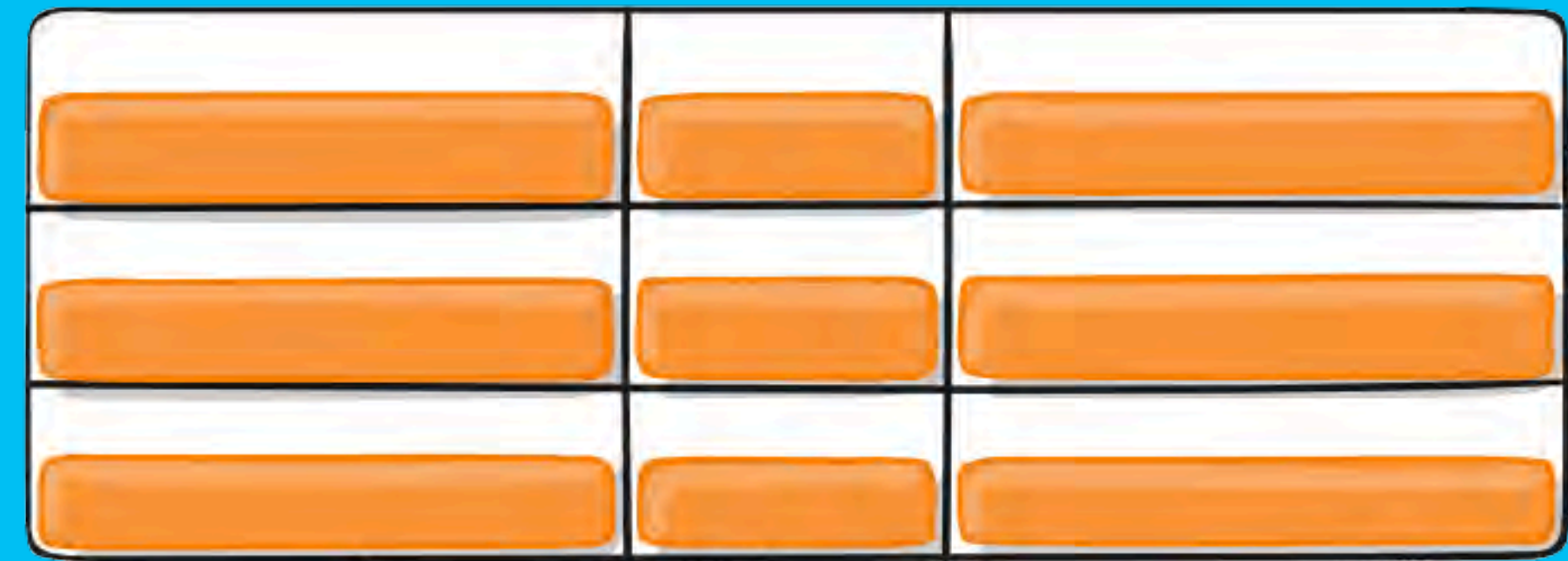
THE PARENT - TEMPLATE

ALIGN-ITEMS

- Aligns grid items along the column axis = vertical positioning
- Values:
 - start
 - end
 - center
 - stretch (default)

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  align-items: end;  
}
```

end



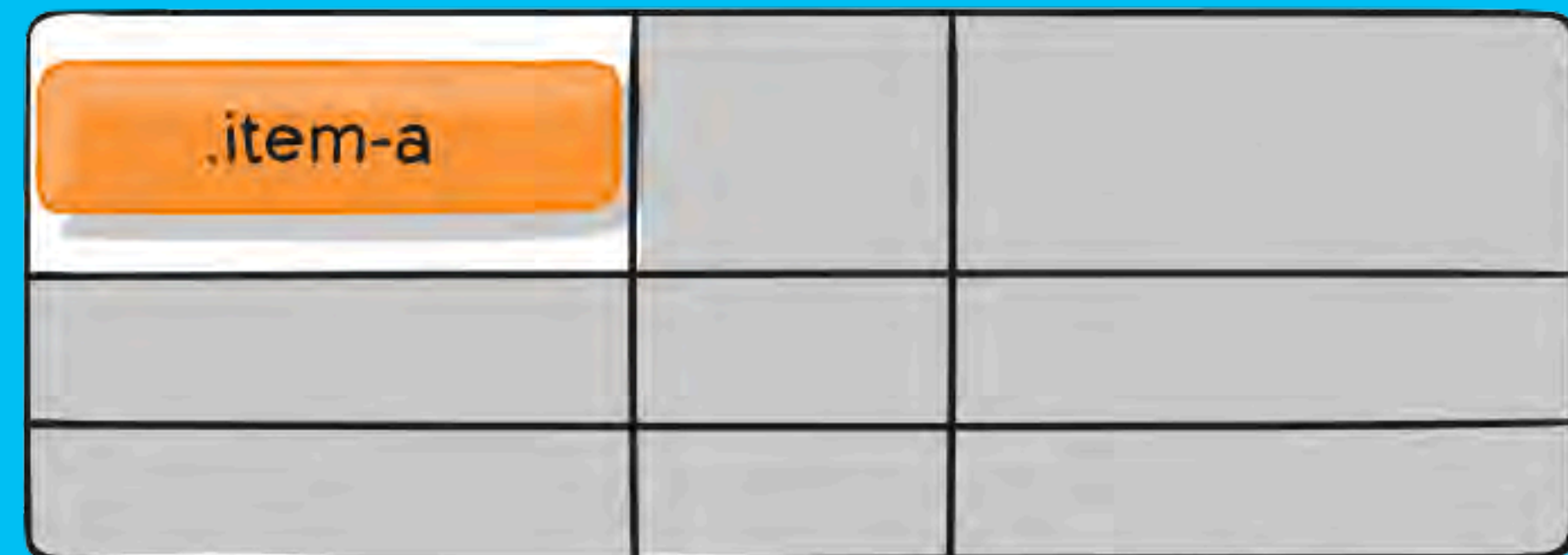
THE PARENT - TEMPLATE

PLACE-ITEMS

- **SHORTHAND:**
- justify-items & align-items
- Position grid in center:
`place-items: center center;`

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  place-items: center stretch;  
}
```

center stretch



THE CHILDREN JUSTIFY-SELF

- Aligns grid item **inside the cell** along the row axis = horizontal
- Values:
 - start
 - end
 - center
 - stretch (default)

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  justify-items: center;  
  
  .grid-item-one {  
    justify-self: end;  
  }  
}
```

Grid Container:



Grid Item:



THE CHILDREN ALIGN-SELF

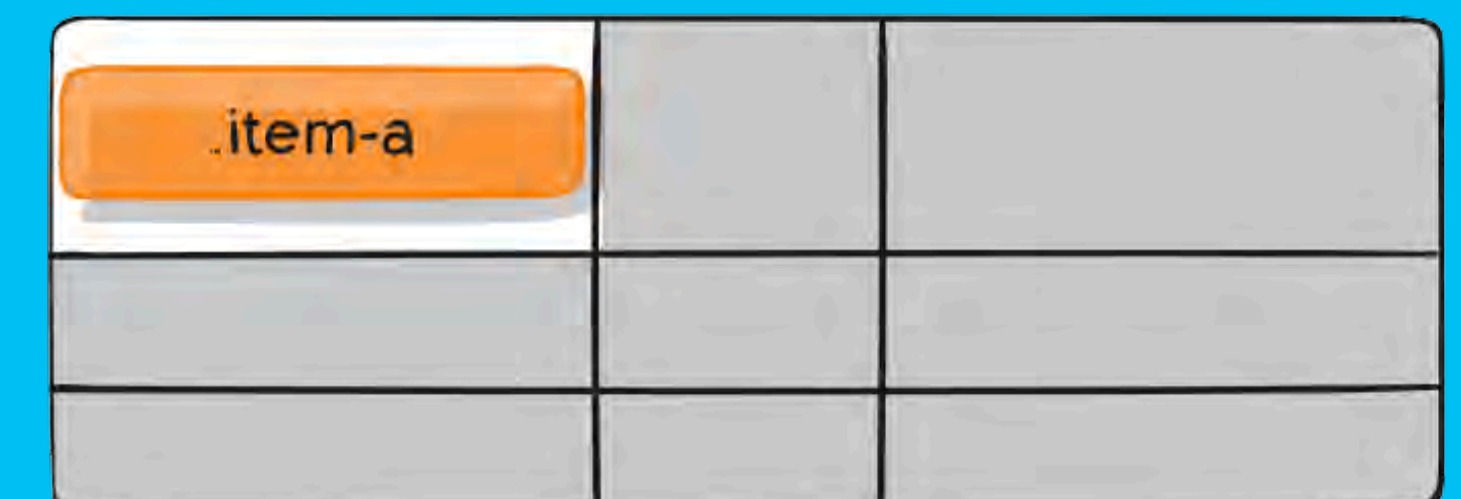
- Aligns grid item **inside the cell** along the column axis = vertical
- Values:
 - start
 - end
 - center
 - stretch (default)

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  align-items: end;  
  
  .grid-item-one {  
    align-self: center;  
  }  
}
```

Grid Container:



Grid Item:

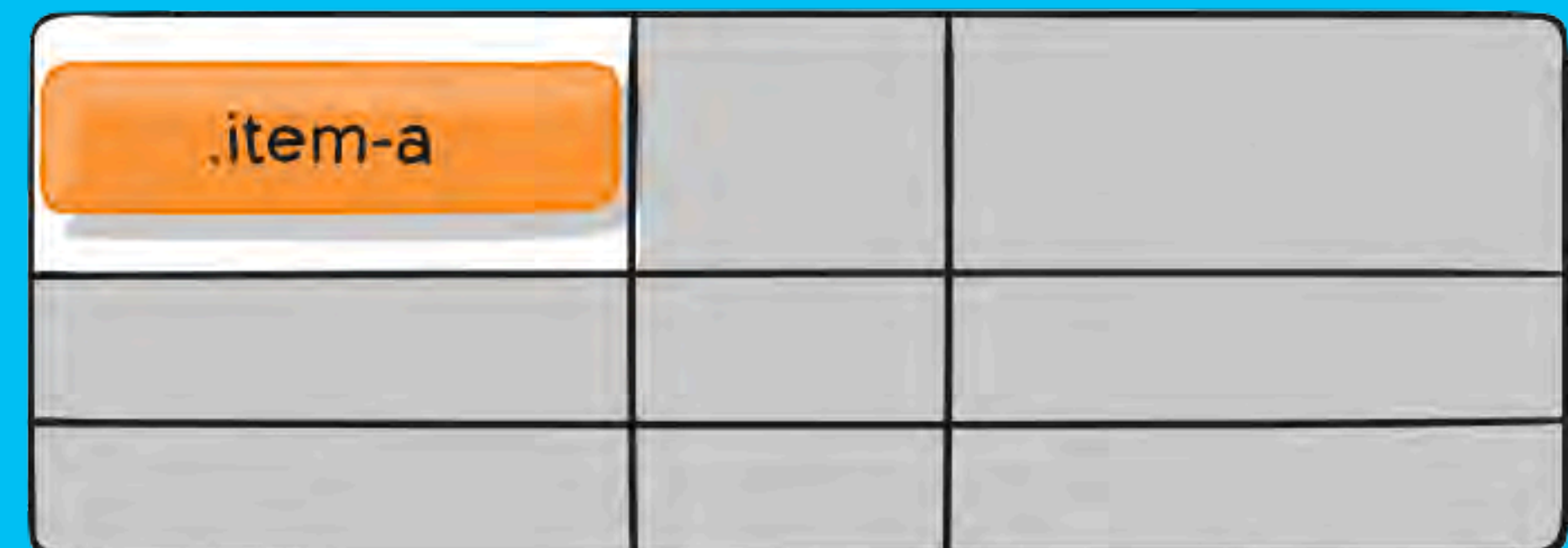


THE CHILDREN PLACE-SELF

- **SHORTHAND:**
- align-self & justify-self
- Values:
 - align-self
 - justify-self
 - auto
- Position cell in center:
place-self: center center;

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  align-items: end;  
  
  .grid-item-one {  
    place-self: center stretch;  
  }  
}
```

center stretch



THE PARENT - TEMPLATE

JUSTIFY-CONTENT

- Aligns grid within grid container along row axis = horizontal positioning
- Values:
 - start
 - end
 - center
 - stretch (default)
 - space-around
 - space-between
 - space-evenly

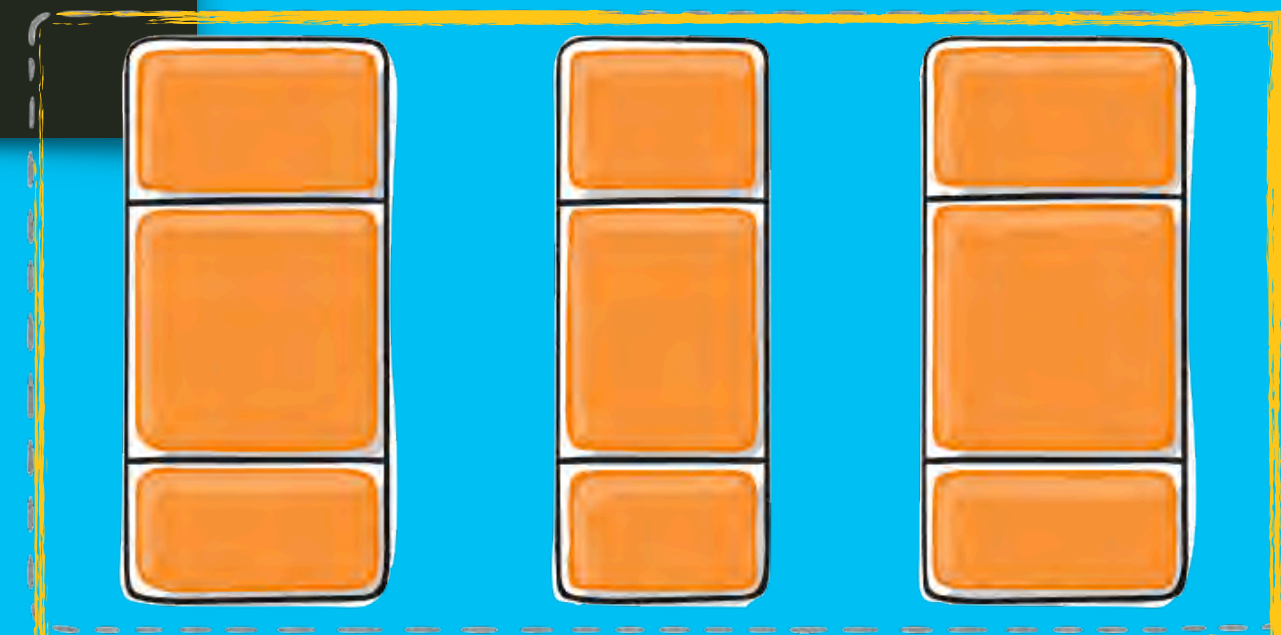
```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  justify-content: end;  
}
```

grid container



```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  justify-content: space-around;  
}
```

grid container



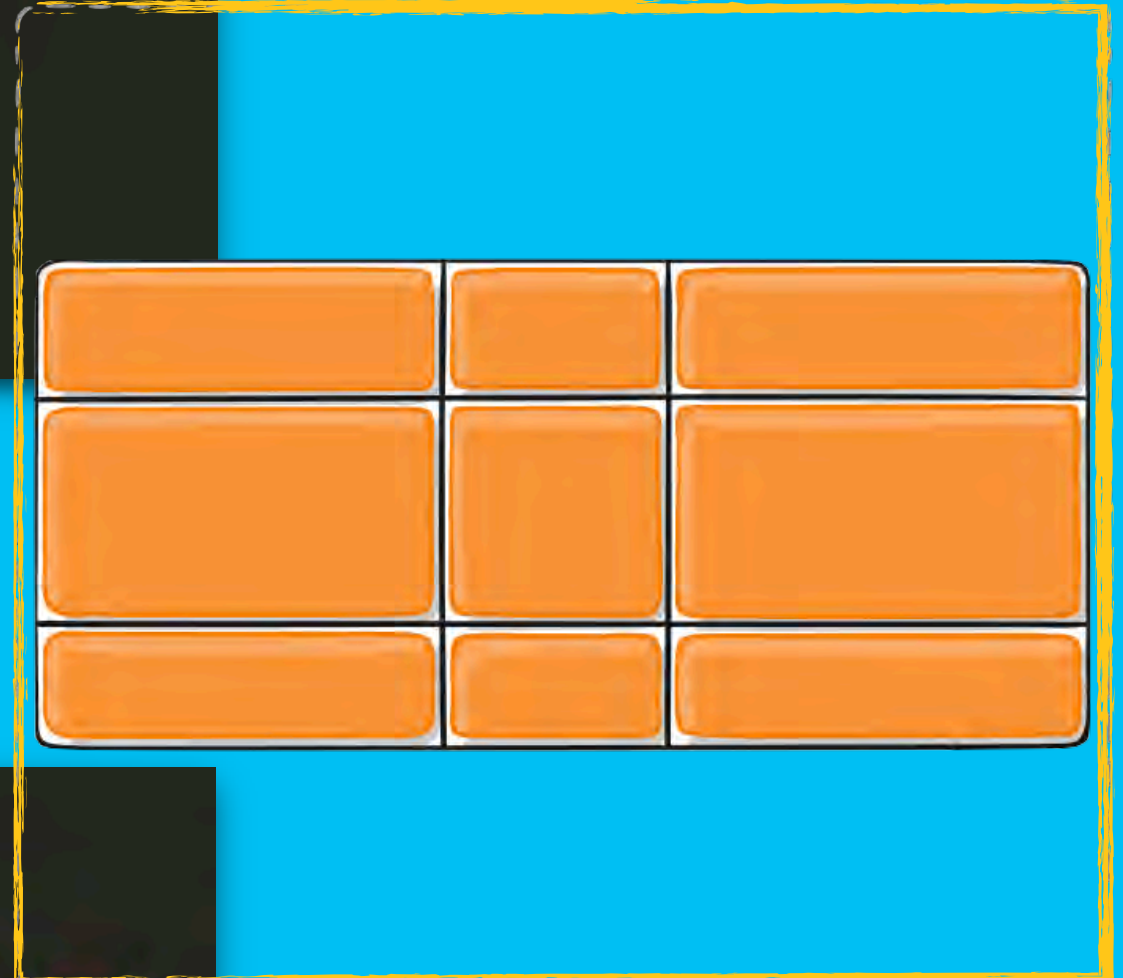
THE PARENT - TEMPLATE

ALIGN-CONTENT

- Aligns grid within grid container along column axis = vertical positioning
- Values:
 - start
 - end
 - center
 - stretch (default)
 - space-around
 - space-between
 - space-evenly

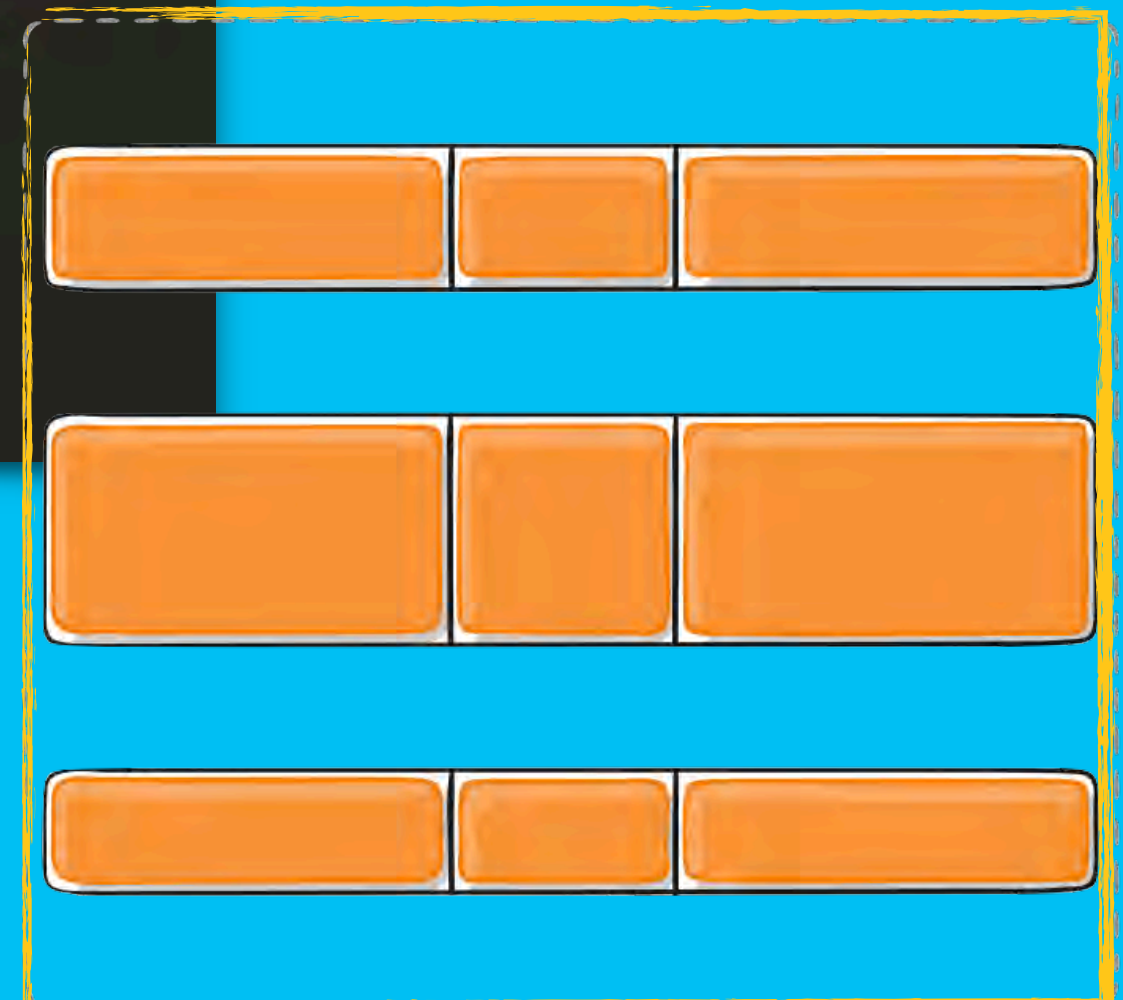
```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  align-content: center;  
}
```

grid container



```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  align-content: space-evenly;  
}
```

grid container



THE PARENT - TEMPLATE

PLACE-CONTENT

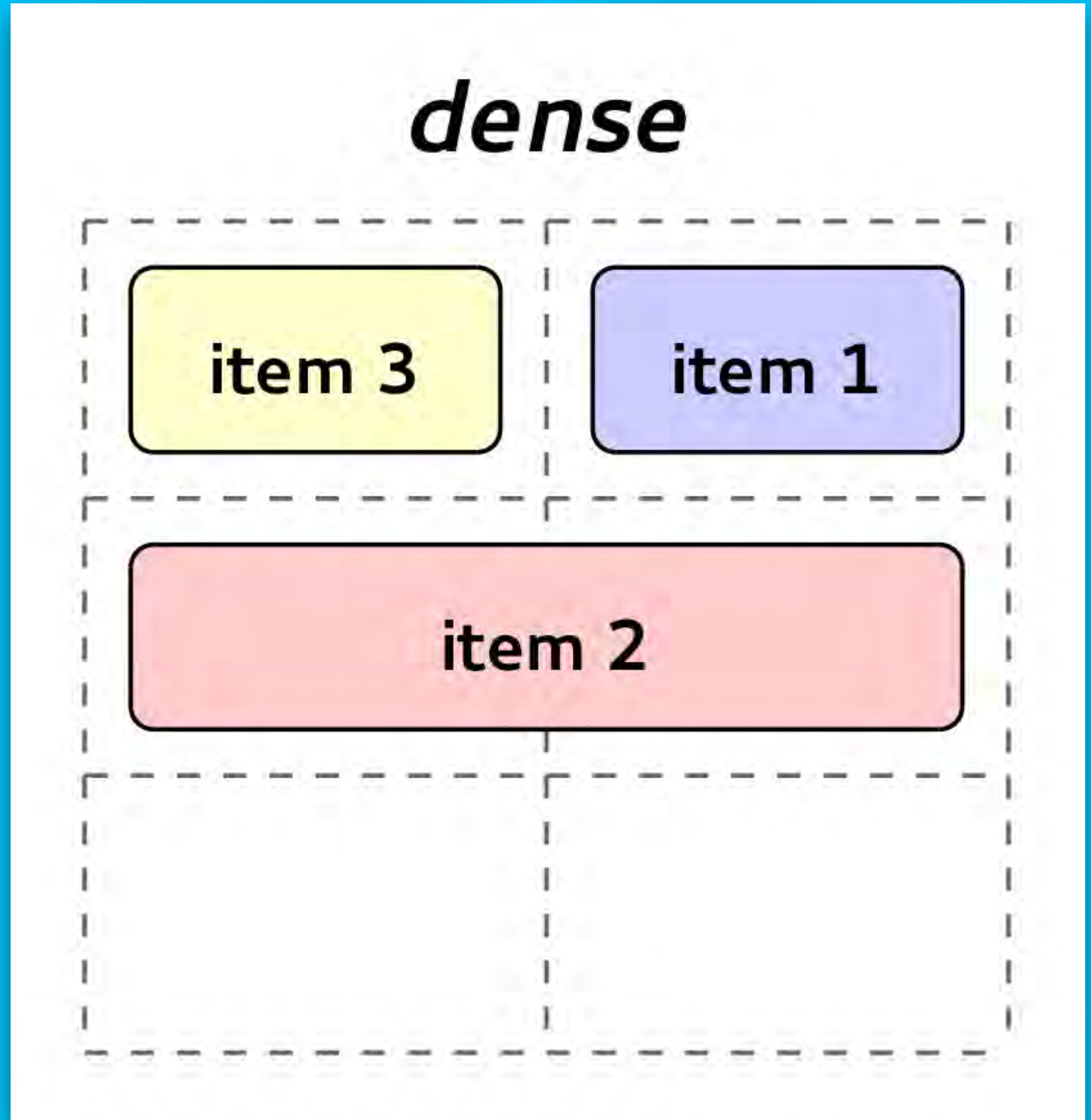
- **SHORTHAND:**
- align-content & justify-content
- Position content in center:
place-content: center center;

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: auto;  
  grid-auto-rows: minmax(200px, 1fr);  
  gap: 1em;  
  
  place-content: center stretch;  
}
```


THE PARENT - TEMPLATE

GRID-AUTO-FLOW

- Like flex-direction
- Values:
 - row (default)
 - column
 - dense —> tells the auto-placement algorithm to attempt to fill in holes earlier in the grid if smaller items come up later



SPECIAL USES:

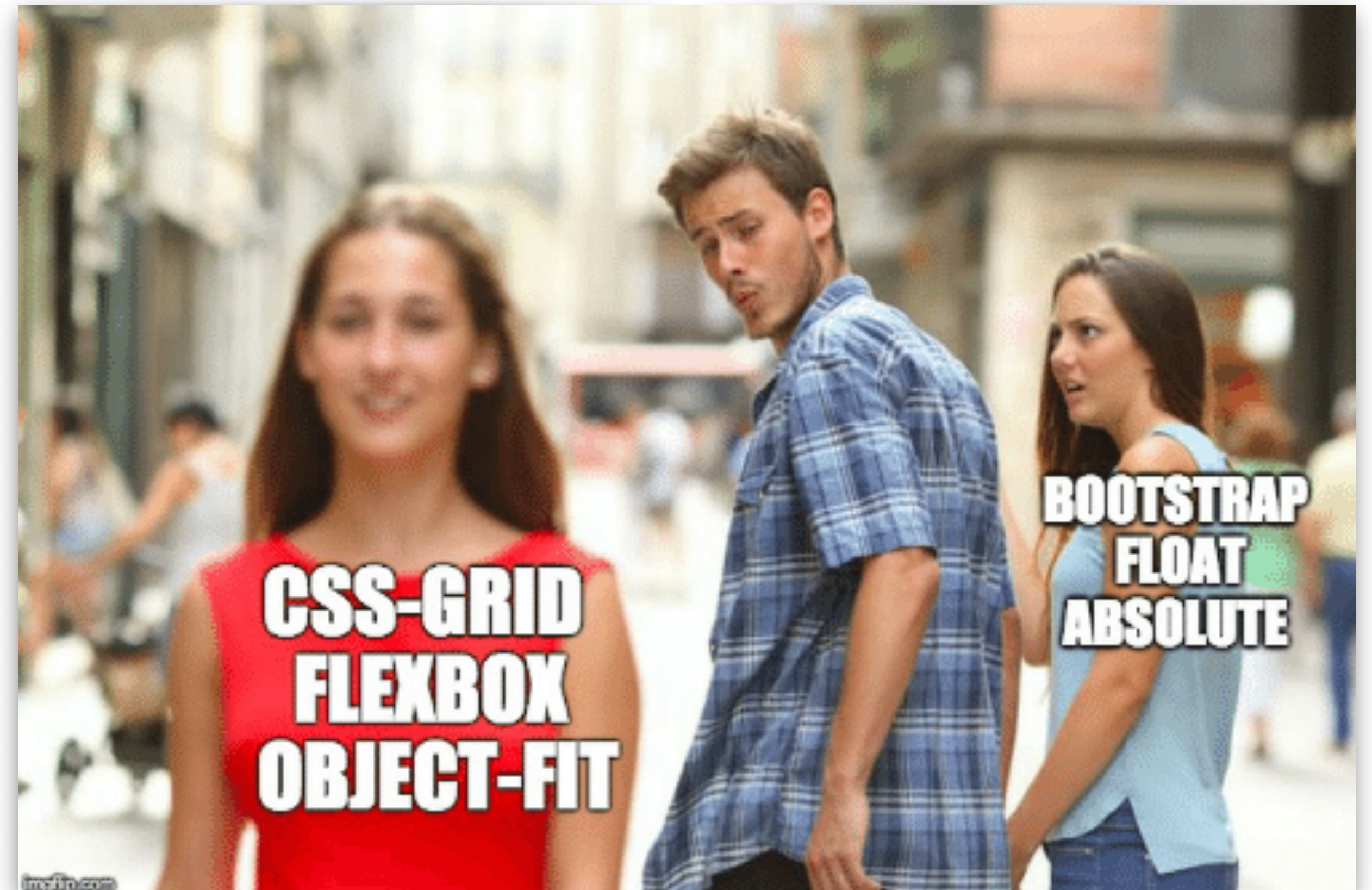
- `repeat(x,y)`: takes two values: `repeat(amount of rows/columns, unit)`
- `span`: instead of specifying end row/column, set how many cells you want the item to span! But only works with positive values: `span 2`
- `grid-auto-fill`: grid does the math for you and essentially makes your grid responsive, can replace media-queries, if used with `minmax`
- `grid-auto-fit`: always ends grid after last item and expands elements inside grid. THE one line of code you should remember!
- `grid-row/column: 1/-1` —> easy use to emulate width: 100% on grid!
- Empty space: simply use a dot (.)
- Create mosaic-grid layouts by overlapping items and using z-index to determine hierarchy
- `order`: Use carefully, it will mess up your accessibility (focus order)

SO WHICH ONE DO I USE?



YOU USE BOTH!

- You can use both Flexbox and Grid in the same project!
- Use Grid for:
 - Building layout first
 - Larger page layouts
 - Controlling rows **and** columns
 - Overlapping elements
- Use Flexbox for:
 - Building content first
 - Small, one-dimensional components
 - Controlling rows **or** columns
 - Dynamic spacing between elements



QUESTIONS?

WHOA, I KNOW

CSS-GRID

memegenerator.net

CSS GRID RESOURCES

- A complete guid to CSS Grid:

<https://css-tricks.com/snippets/css/complete-guide-grid/>

- Grid Garden:

<https://cssgridgarden.com/>

- Grid by Example:

<https://gridbyexample.com/>

REAL LIFE EXAMPLES

- <https://codepen.io/michellejames/pen/QWGGvLp?editors=0110>
- <https://codepen.io/girlgeek/pen/OgqBgj?editors=1100>
- <https://codepen.io/andybarefoot/pen/wrXvLj>
- <https://codepen.io/winkerVSbecks/pen/KmoxJp>

HOMework - CSS GRID

- Practice your new CSS Grid knowledge by completing the CSS Grid Garden Tutorial! Take a screenshot at the end by clicking on the level, revealing 28 checkmarks so I can be sure you did all levels and submit it along with homework! I want a screenshot from your entire desktop, open tabs, date, time - the whole shebang!
- Use the provided Adobe XD file to recreate the layout using CSS Grid.
- Export all the images you need from the file and optimize them with the tools and techniques we've already learned, if needed.
- Add an additional design for the tablet/small desktop breakpoint. It should fit seamlessly between mobile and desktop layout.
- As usual, the page must be fully responsive. Please use Sass in your CSS.
- Submit a link to your GitHub repo.

FIN