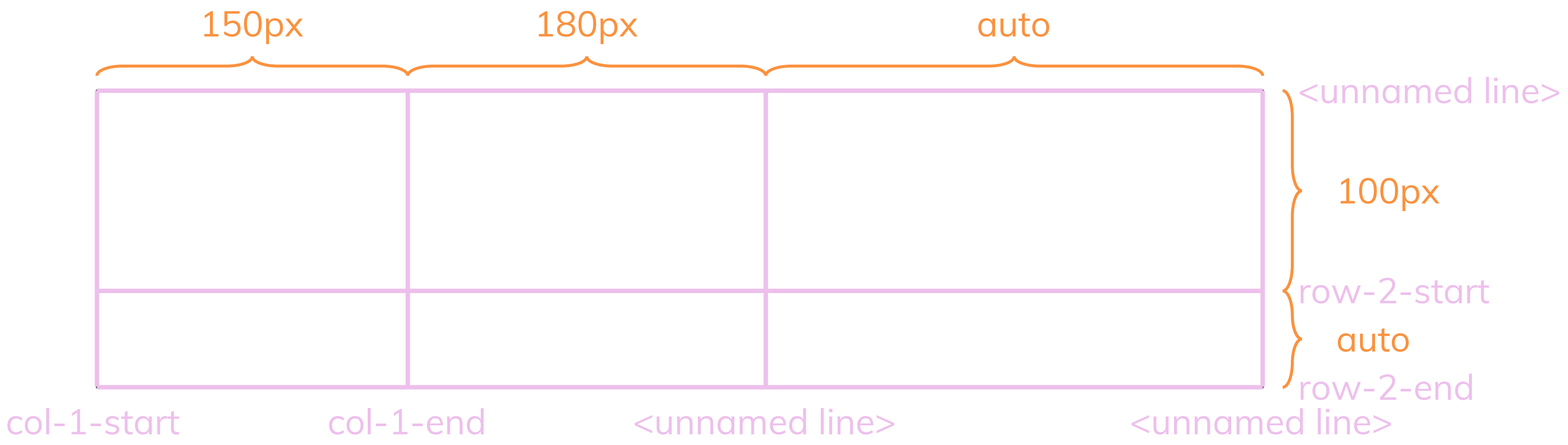
Grid Templates

```css
.grid-contai
  height: 140px;
  display: grid;
  grid-template-columns: [col-1-start] 150px [col-1-end] 180px auto;
  grid-template-rows: auto [row-2-start] 100px [row-2-end];
}
```

Turn <div> into a grid

Assign name to line

Assign width/height

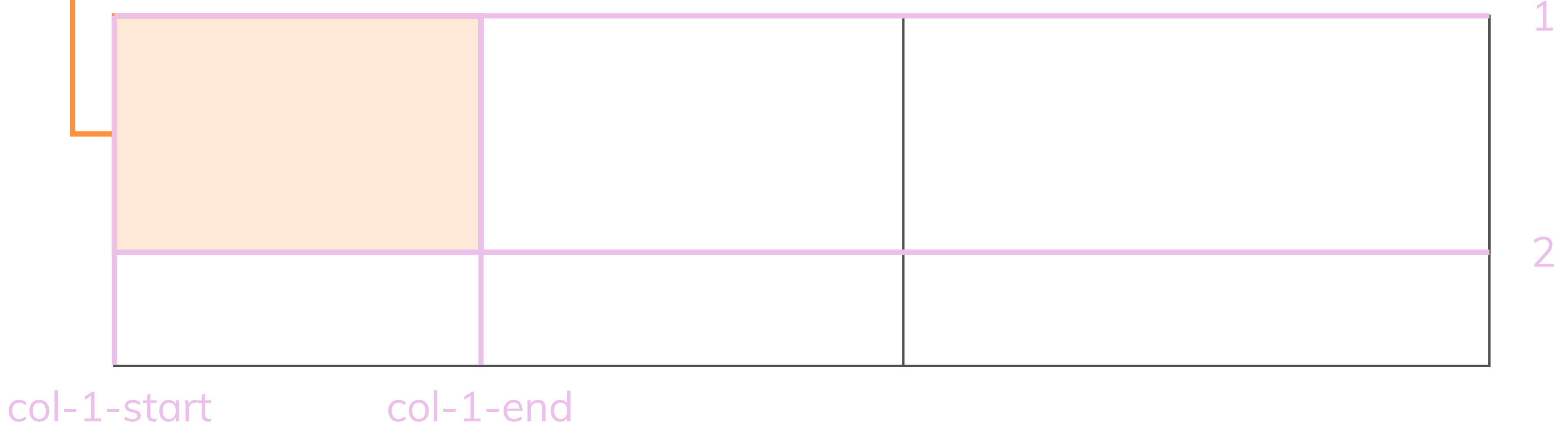


From a Grid Cell Perspective

```
.cell-1 {
  grid-column-start: col-1-start;
  grid-column-end: col-1-end;
  grid-row-start: 1;
  grid-row-end: 2;
}
```

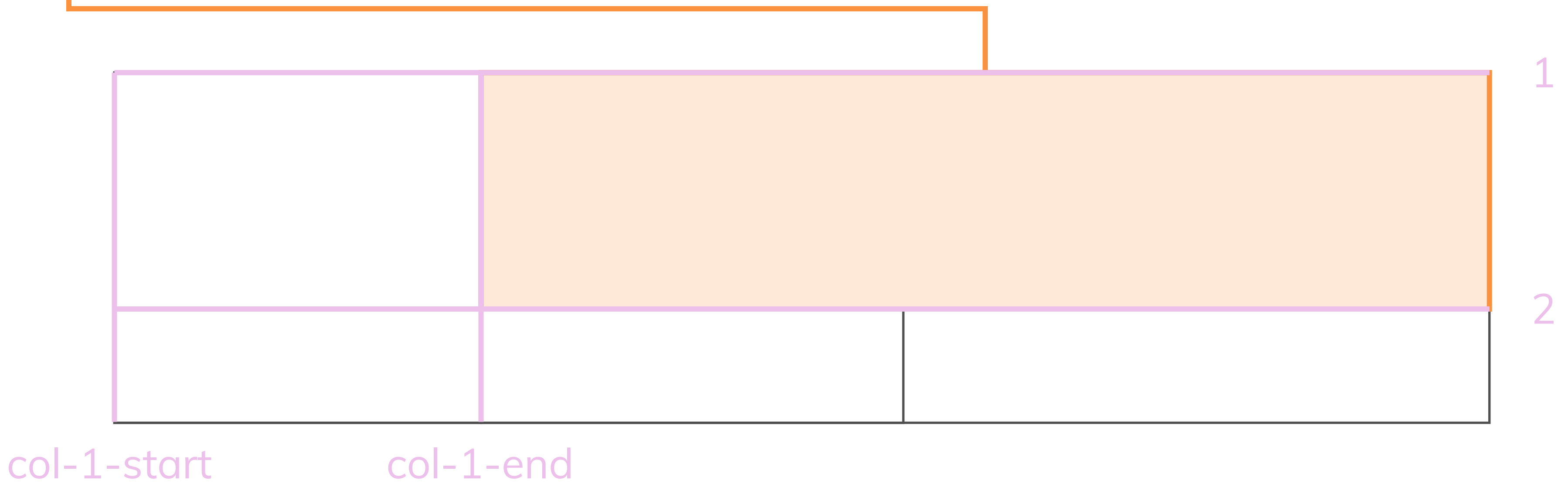
Shorthand: `grid-column: col-1-start / col-1-end`

Refer to line names (if set) or line numbers



From a Grid Cell Perspective

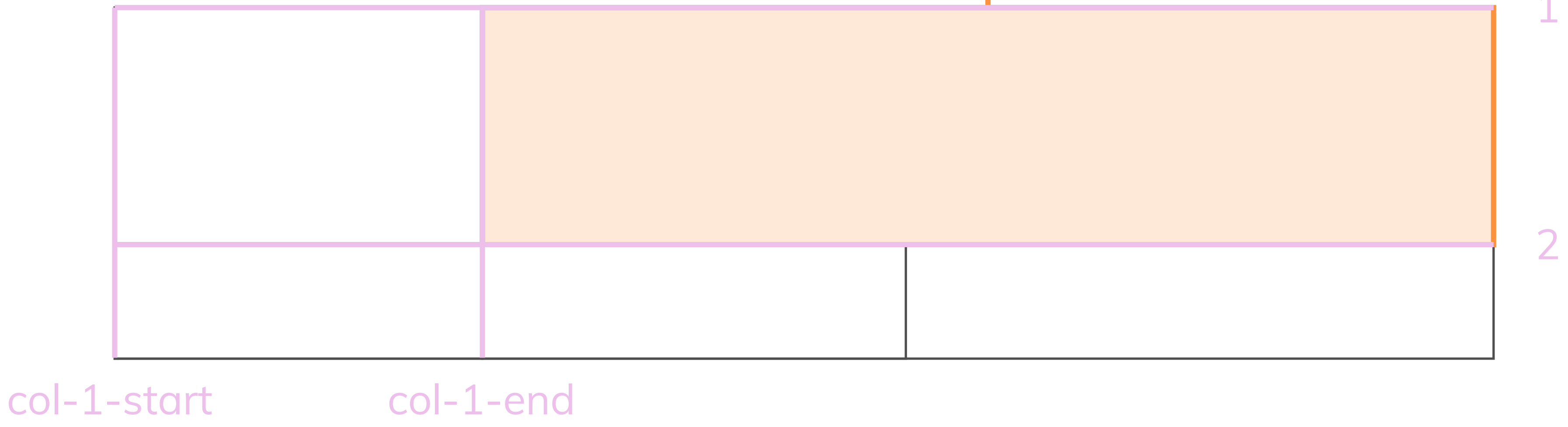
```css
.cell-2 {
  grid-column-start: col-1-end;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 2;
}
```



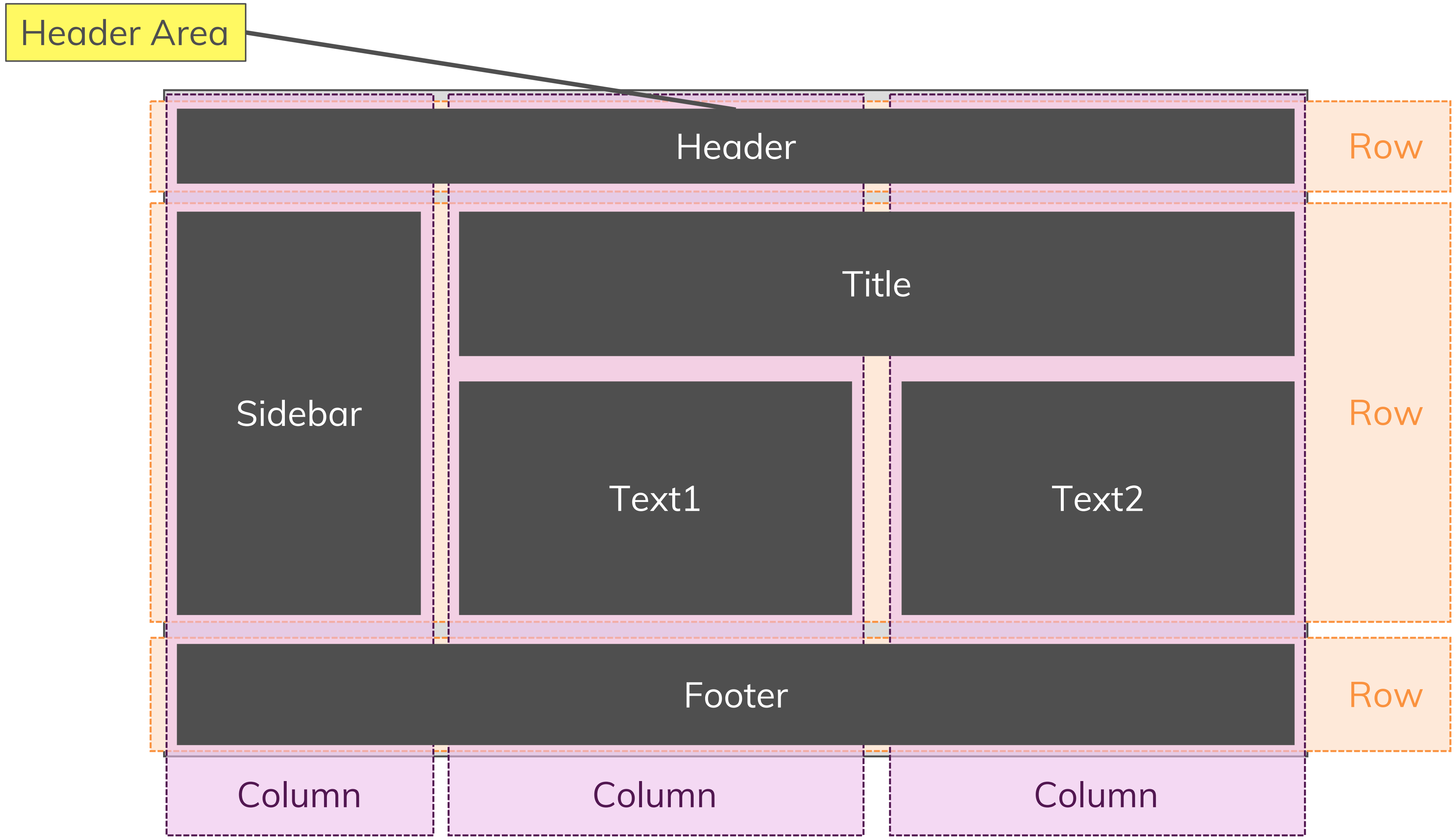
An Alternative Way

```
.cell-2 {
  grid-column-start: col-1-end;
  grid-column-end: span 2;
  grid-row-start: 1;
  grid-row-end: 2;
}
```

By the Way: Overlapping is allowed, control stacking via `z-index`

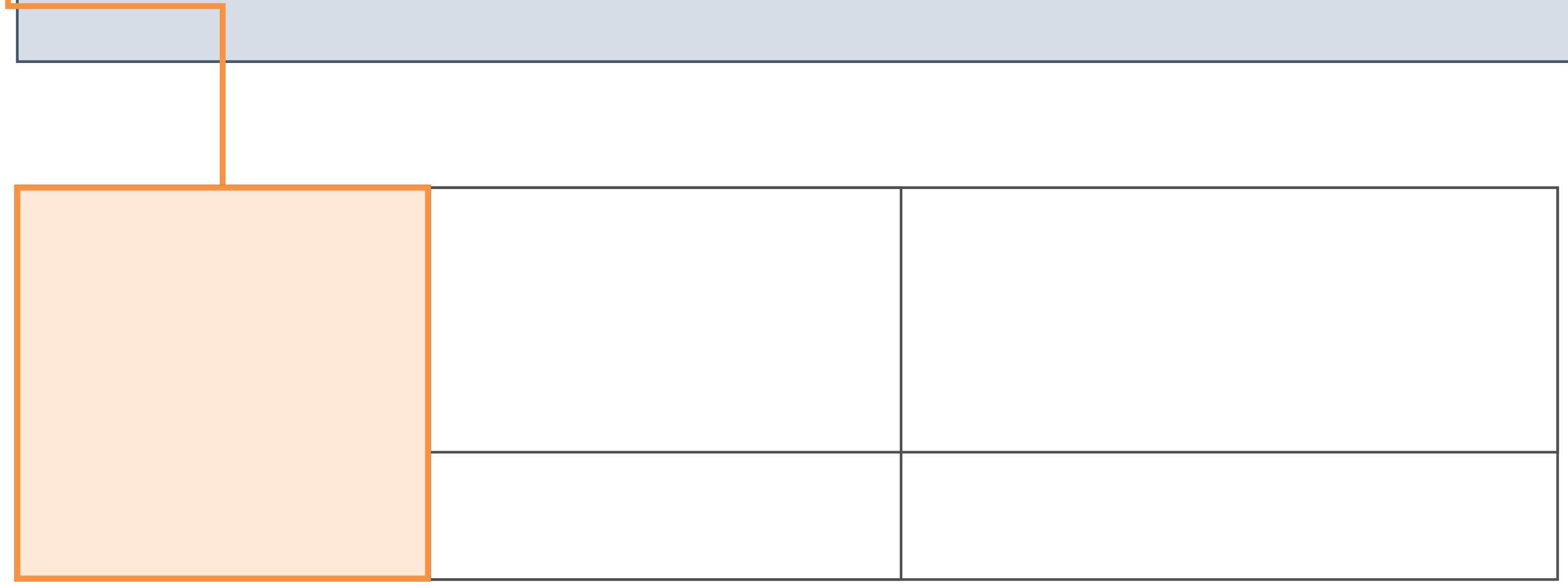


Grid Areas

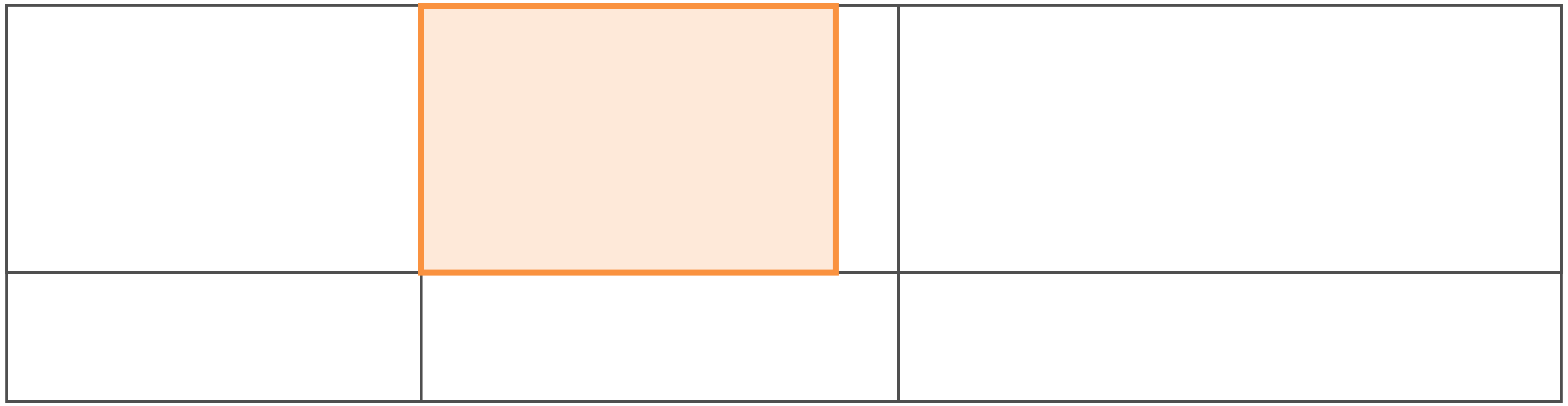

Header Area
Header
Row
Sidebar
Title
Text1
Text2
Row
Footer
Row
Column
Column
Column

From a Grid Cell Perspective

```css
.sidebar {
  grid-area: sidebar;
}
```

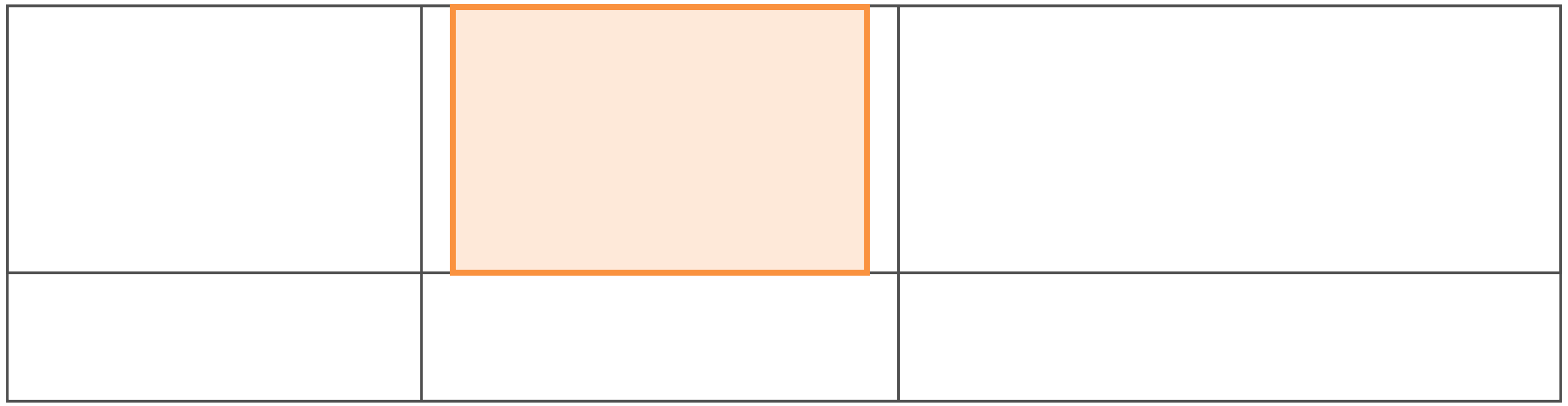


Grid Alignment – Horizontal Start



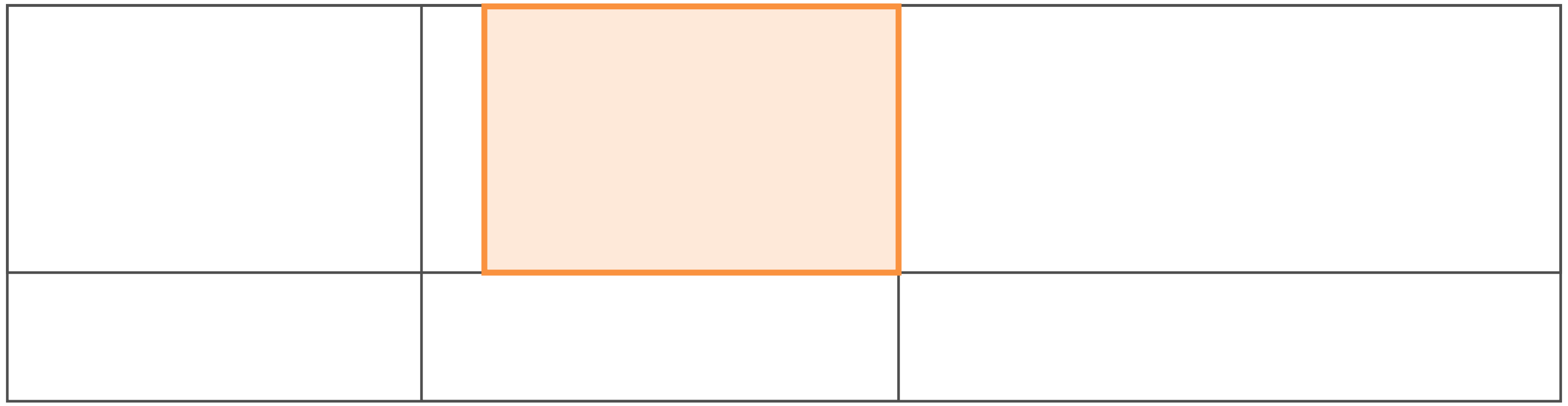
```
.grid-container {
  justify-items: start;
}
```

Grid Alignment - Horizontal Center



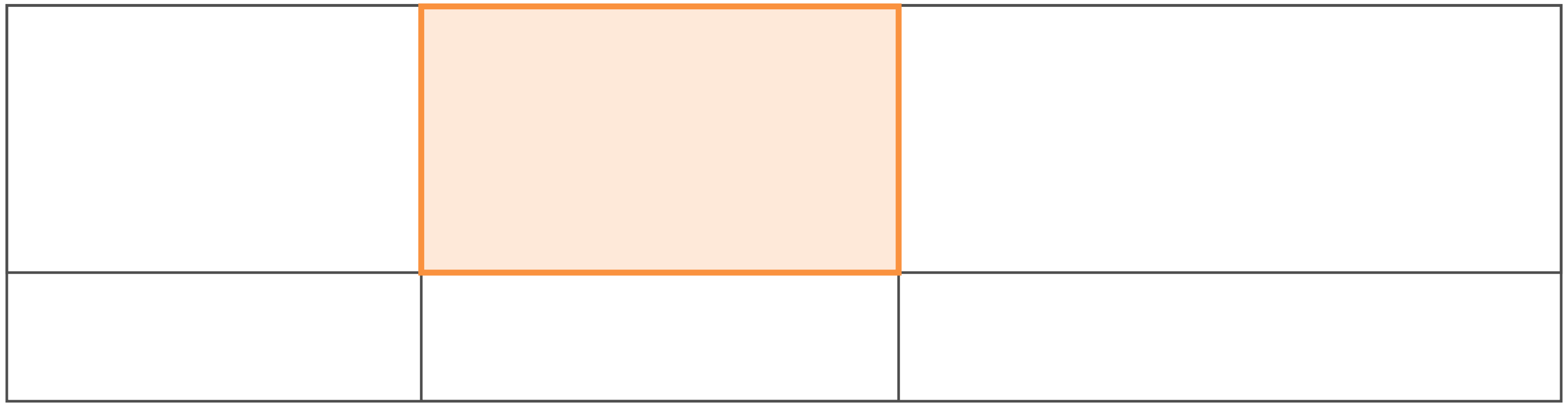
```
.grid-container {
  justify-items: center;
}
```

Grid Alignment - Horizontal End



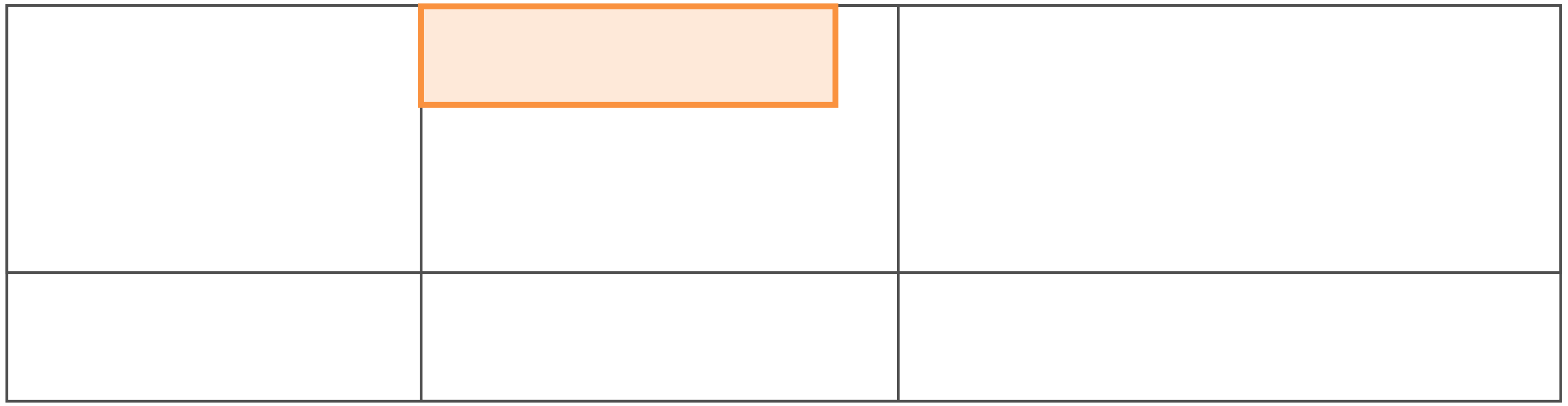
```
.grid-container {
  justify-items: end;
}
```

Grid Alignment - Horizontal Stretch



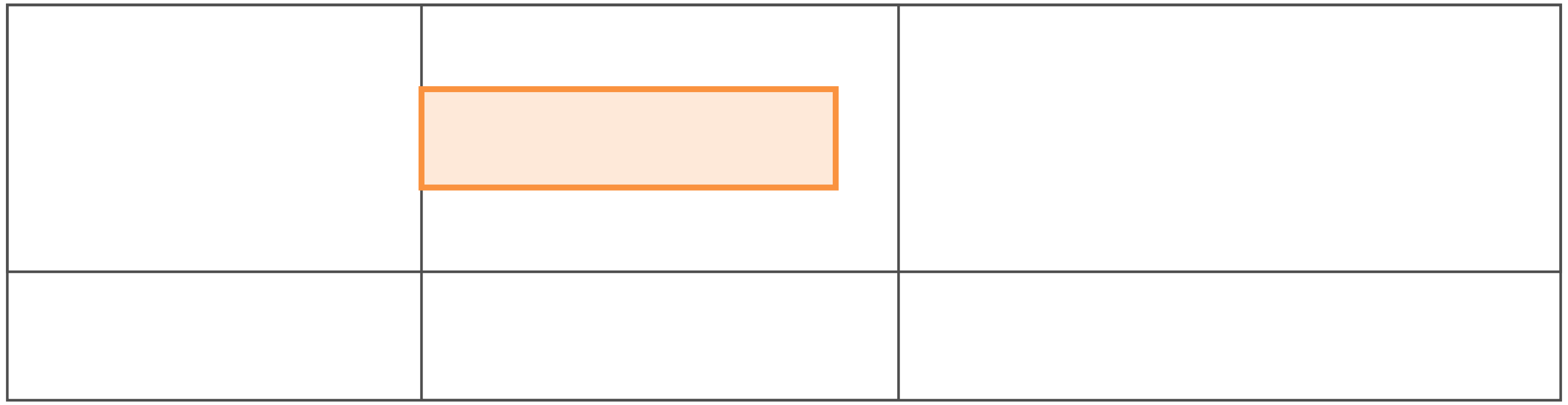
```css
.grid-container {
  justify-items: stretch;
}
```

Grid Alignment – Vertical Start



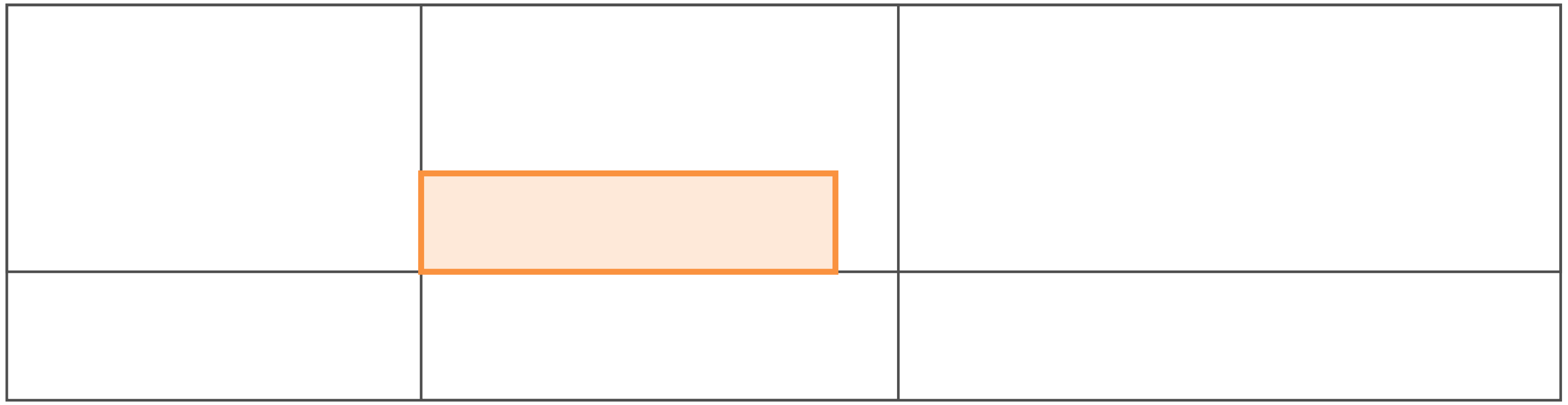
```css
.grid-container {
  align-items: start;
}
```

Grid Alignment - Vertical Center



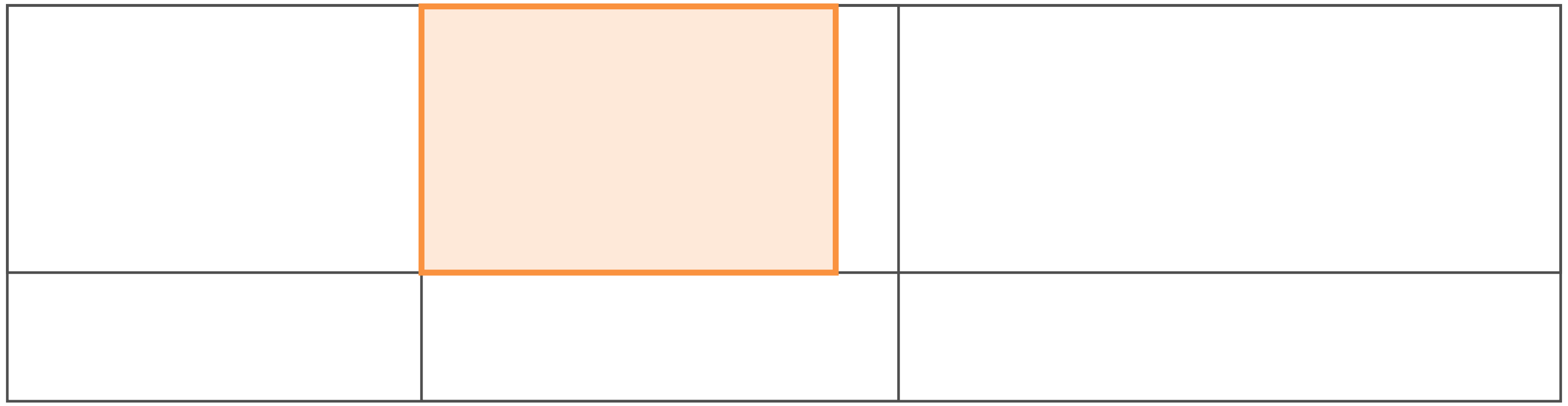
```
.grid-container {
  align-items: center;
}
```

Grid Alignment - Vertical End



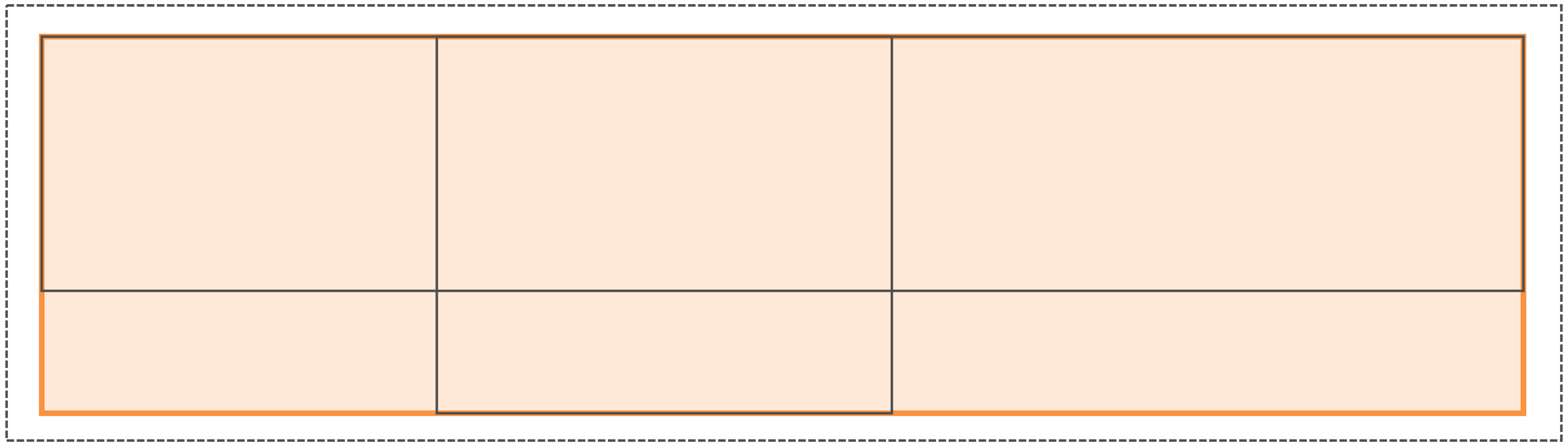
```
.grid-container {
  align-items: end;
}
```

Grid Alignment - Vertical Stretch



```css
.grid-container {
  align-items: stretch;
}
```

Grid Alignment – Align Grid Itself

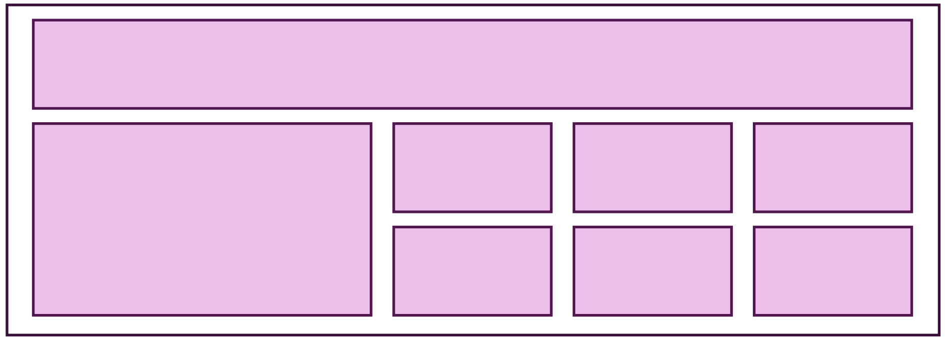


```
.grid-container {
  justify-content: start | end | center | stretch | space-around | space-
between | space-evenly;
  align-content: start | end | center | stretch | space-around | space-
between | space-evenly;
}
```

CSS Grid vs Flexbox

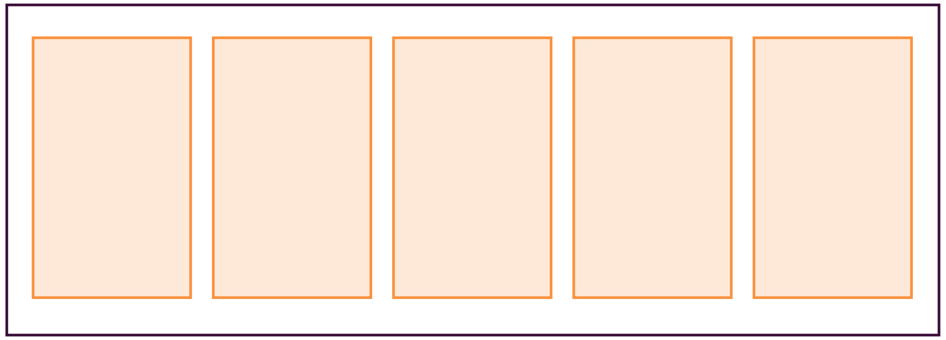
CSS Grid

Two-dimensional Positioning



CSS Flexbox

One-dimensional Positioning



Summary

Creating a Grid

- `display: grid` creates a grid where child elements are automatically placed in rows
- This default can be overwritten with `grid-auto-flow` (and then also `grid-auto-rows` or `grid-auto-columns`)
- Use `grid-gap` to add gaps between columns and rows

Defining the Grid Structure

- You define columns and/ or rows explicitly via `grid-template-columns`/ `grid-template-rows`
- Use `repeat(times, size)` to create multiple columns or rows with ease
- Use `auto-fill`/ `auto-fit` to derive the number of columns automatically
- Use `minmax` for dynamic sizing

Placing Elements

- Position elements in the grid via `grid-row` and/ or `grid-column`
- Use `span X` to span an element over multiple columns or rows
- Use line numbers, line names or named areas

Aligning Elements

- Align grid items via `justify-items` (X-axis) and `align-items` (Y-axis)
- Align the entire grid content via `justify-content` (X-axis) and `align-content` (Y-axis)