CSS3 Grid & Flexbox

Grid (2D) ⟶ Page Layout

Flexbox (1D) ⟶ Contents

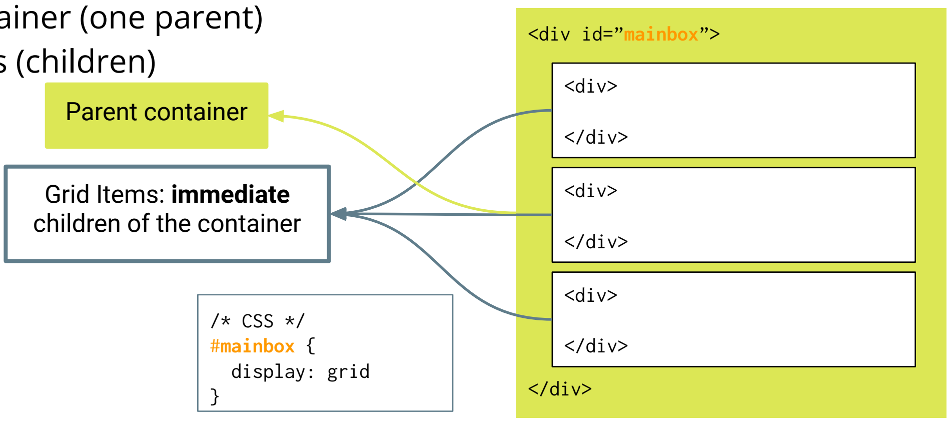
Which one?

- Use CSS Grid to organize 2D layout of major elements (“macro”)
- Use CSS Flexbox to organize contents within an element (“micro”)
- The scale of macro/micro is subjective
- Resources:
    - A Complete Guide to Grid
    - A Complete Guide to Flexbox

CSS Grid

Elements of CSS Grid

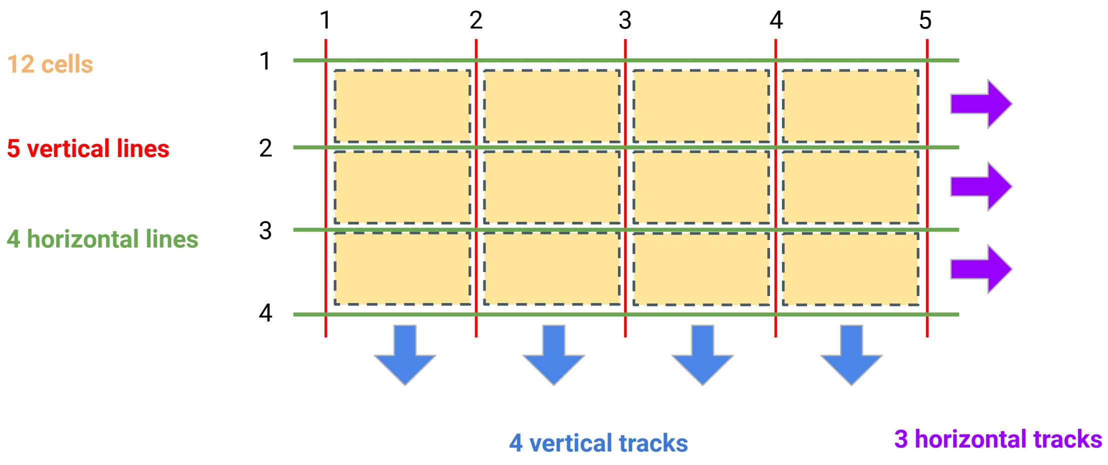
- Organize (page) layout into a MxN flexible rectangular spaces (cells)
- Grid Container (one parent)
- Grid Items (children)



```
<div id="mainbox">
  <div>

  </div>
  <div>

  </div>
  <div>

  </div>
</div>
```

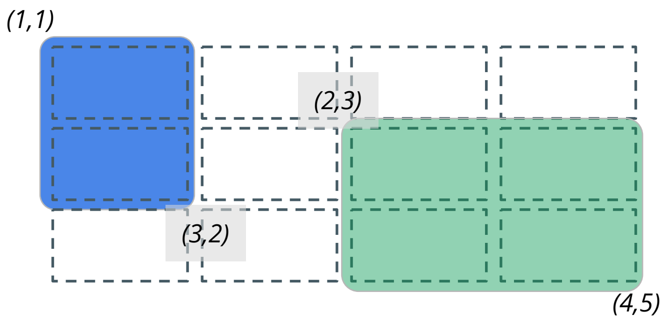
```
/* CSS */
#mainbox {
  display: grid
}
```

Grid Details: Lines & Cells & Tracks



Grid (Rectangular) Areas

- Items may occupy multiple cells

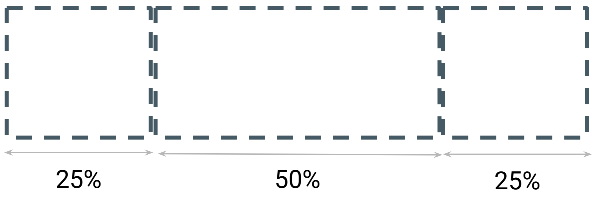


```
#blueBox {
  grid-area: 1/1/3/2;
}
```

```
#greenCorner {
  grid-area: 2/3/4/5;
}
```

Grid Template (Rows|Columns)

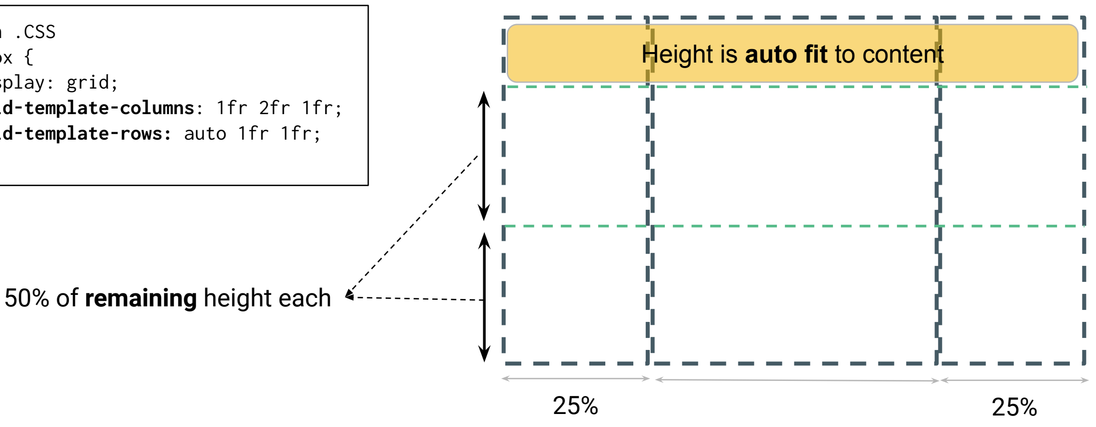
```
// in .CSS
.mybox {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;
}
```



Unit	Description
`auto`	Just enough to fit content
`fr`	Proportions of the available parent space (width or height)
`%`	Percentage of the available parent space
`px, em, cm, …`	(Minimum size)

Grid Template

```
// in .CSS
.mybox {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;
  grid-template-rows: auto 1fr 1fr;
}
```

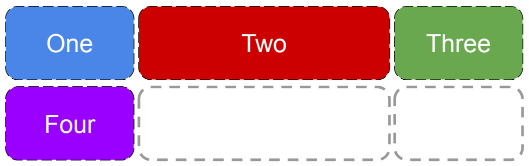

Height is **auto fit** to content
50% of **remaining** height each
25%
25%

Default Placement

- Default placement: children fill the cells left-to-right, top-to-bottom

```
<div id="mybox">
  <span class="blue">One</span>
  <span class="red">Two</span>
  <span class="green">Three</span>
  <span class="purple">Four</span>
</div>
```

```
#mybox {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;
}
```

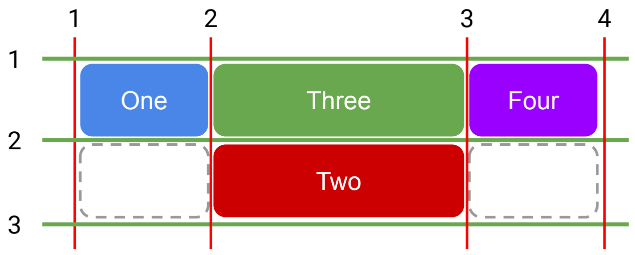


JS Fiddle

Explicit positioning by “coordinates”

```
<div id=”mybox”>
  <span class=”blue”>One</span>
  <span class=”red”>Two</span>
  <span class=”green”>Three</span>
  <span class=”purple”>Four</span>
</div>
```

```
#mybox {
  display: grid;
  grid-template-rows: 1fr 1fr;
  grid-template-column: 1fr 2fr 1fr;
}

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

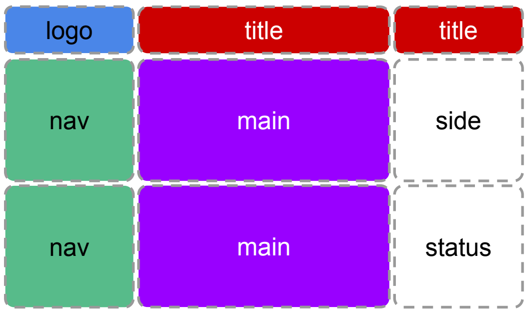


JS Fiddle

Explicit Positioning by Area Names

```
<div id=”mybox”>
  <span class=”blue”>Logo</span>
  <span class=”red”>Title</span>
  <span class=”green”>Nav</span>
  <span class=”purple”>Main</div>
</div>
```

```
#mybox {
  display: grid;
  grid-template-rows: auto 1fr 1fr;
  grid-template-columns: 1fr 2fr 1fr;
  grid-template-areas:
    “logo title title”
    “nav main side”
    “nav main status”;

}
```



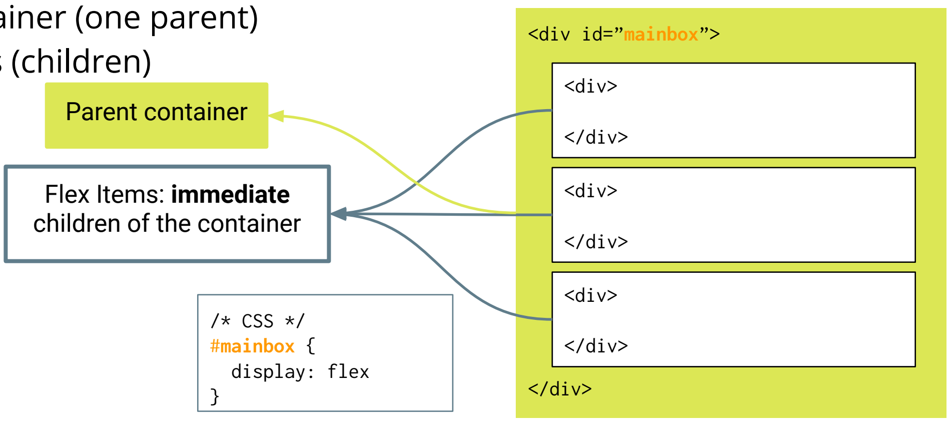
JSFiddle

CSS Flexbox (1D)

Reference: A complete Guide to Flexbox

Elements of CSS Flexbox

- Organize contents into a horizontal/vertical flexible box
- Flex Container (one parent)
- Flex Items (children)



```
<div id="mainbox">
  <div>
  </div>
  <div>
  </div>
  <div>
  </div>
</div>
```

```
/* CSS */
#mainbox {
  display: flex
}
```

Traditional Box vs. FlexBox

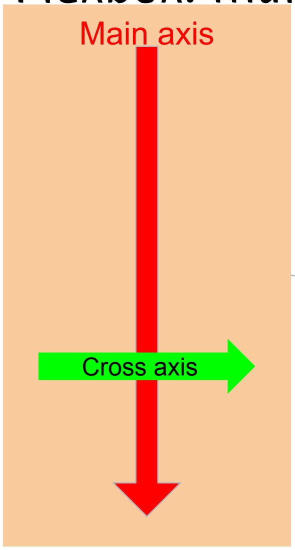
Traditional layout "tricks"

- display: (block|inline)
- float: (left|right)
- position: (fixed|absolute|relative)
- "grid" approach using <table>

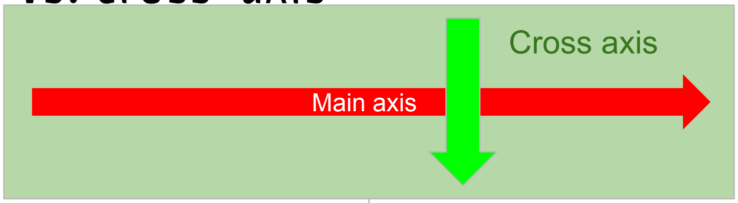
Modern approach

- **Alignment** among elements
- **Distribute** space between elements
- **Shrinkable/expandable** boxes (in both horizontal and vertical directions)
- Flex container (one parent)
- Flex items (children)
- **Main-axis** vs **cross-axis**

Flexbox: main-axis vs. cross-axis



```
// column oriented box
#sample1 {
  display: flex;
  flex-direction: column;
}
```



```
// row oriented box
#sample2 {
  display: flex;
  flex-direction: row;
}
```

JS Fiddle

Flex Container Properties

- `display: flex | inline-flex`
- `flex-direction: row | row-reverse | column | column-reverse`
- `flex-wrap: nowrap | wrap | wrap-reverse`
- `justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly`
- `align-items: flex-start | flex-end | center | stretch | baseline`

Flex containers vs. Flex items

- display: (flex | inline-flex)
- flex-direction (define main-axis)
- flex-wrap (how items respond to resize)
- justify-content (placement of items along the **main-axis**)
- align-items (placement of items along the **cross-axis**)
- align-content: how to distribute lines along the cross-axis when there is extra space

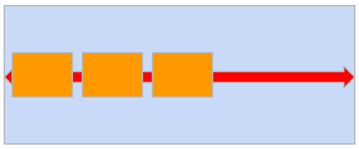
- order: override relative orders
- flex-grow: how items expand to fill up available extra space
- flex-shrink: disable/enable shrinking of elements when parent is shrunk
- align-self: override parent’s align-items

Flexbox: Justification vs. Alignment

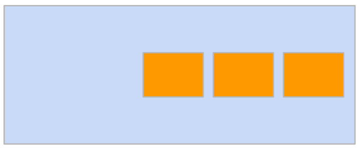
Property	Use by	Purpose
`justify-content`	parent	Placement of the entire contents along the major axis
`align-items`	parent	Placement of individual children along the minor axis
`align-content`	parent	Placement of the entire contents along the minor axis
`align-self`	child	Override the parent align-items property by a child

Justify-content (horizontal box)

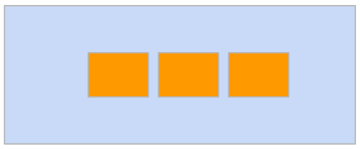
`justify-content: flex-start`



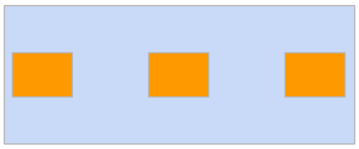
`justify-content: flex-end`



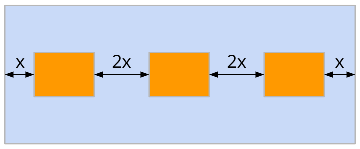
`justify-content: center`



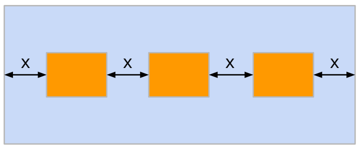
`justify-content: space-between`



`justify-content: space-around`

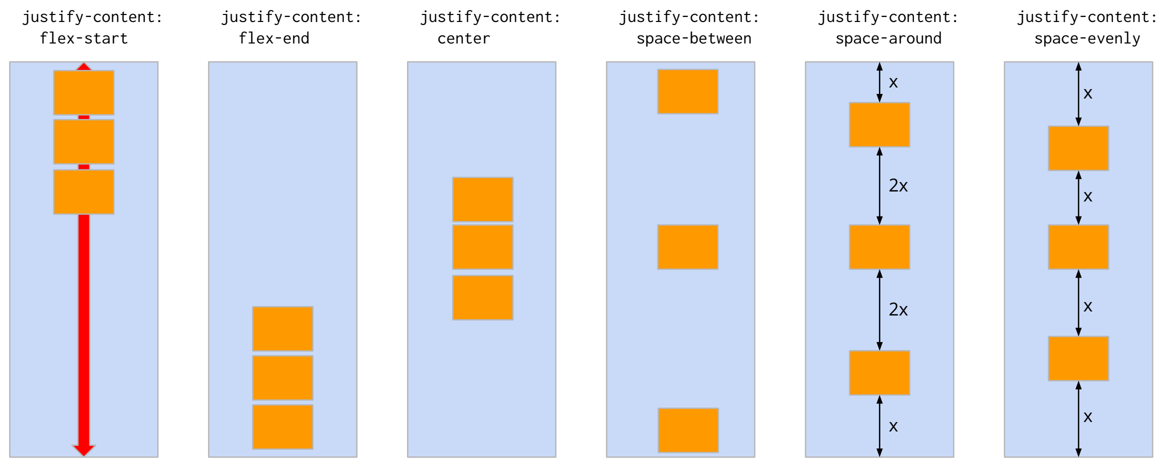


`justify-content: space-evenly`



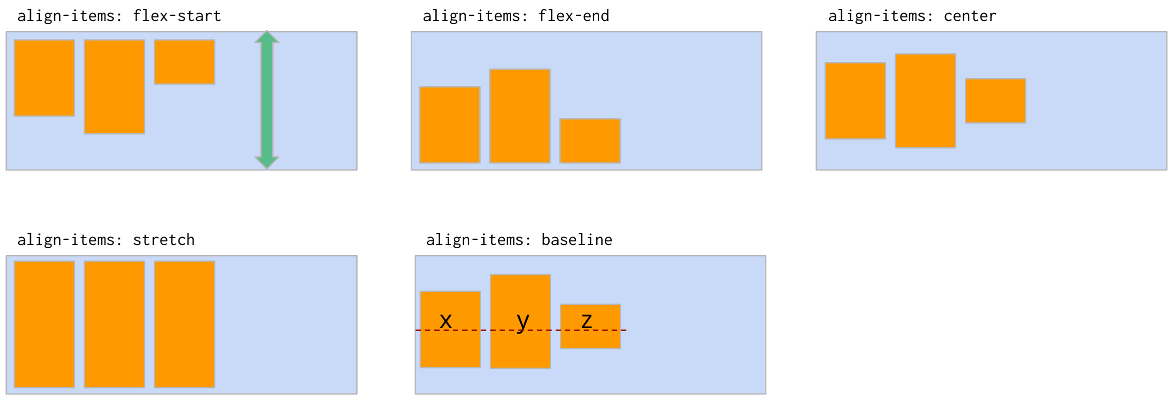
JS Fiddle

Justify-content (vertical box)


justify-content: flex-start
justify-content: flex-end
justify-content: center
justify-content: space-between
justify-content: space-around
x
2x
2x
x
justify-content: space-evenly
x
x
x
x

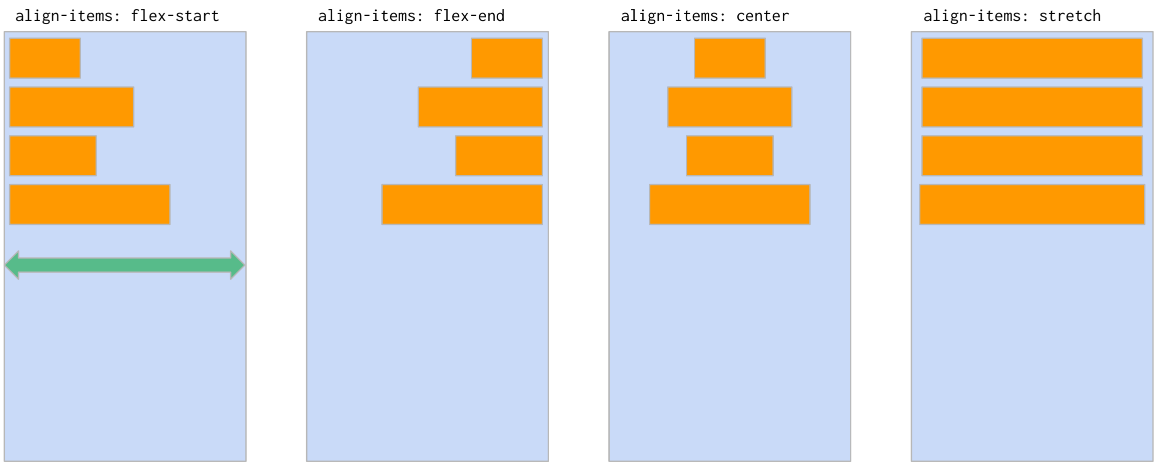
JS Fiddle

Align-items (horizontal box)



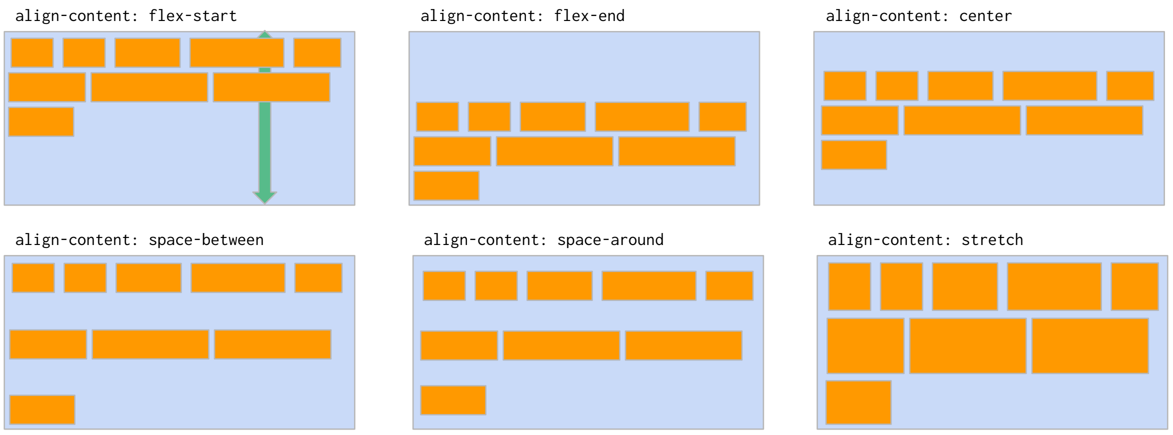
JS Fiddle

Align-items (vertical box)

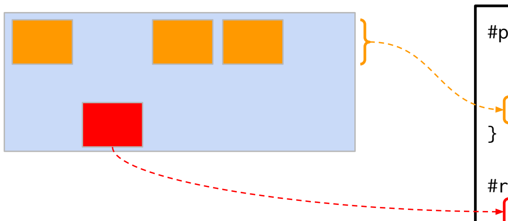


JS Fiddle

Align-content (horizontal box)


align-content: flex-start
align-content: flex-end
align-content: center
align-content: space-between
align-content: space-around
align-content: stretch

Align-self (horizontal box)



```
#parent-box {
  display: flex;
  flex-direction: row;
  align-items: flex-start;
}

#red-box {
  align-self: flex-end;
}
```

JS Fiddle