CIS 371 Web Application Programming

CSS3 Grid & Flexbox

GRAND VALLEY STATE UNIVERSITY®

Lecturer: Dr. Yong Zhuang

Based on the original version by Professor Hans Dulimarta.

Grid & Flexbox

Grid (2D) → Page Layout

Flexbox (1D) → Contents

Which one?

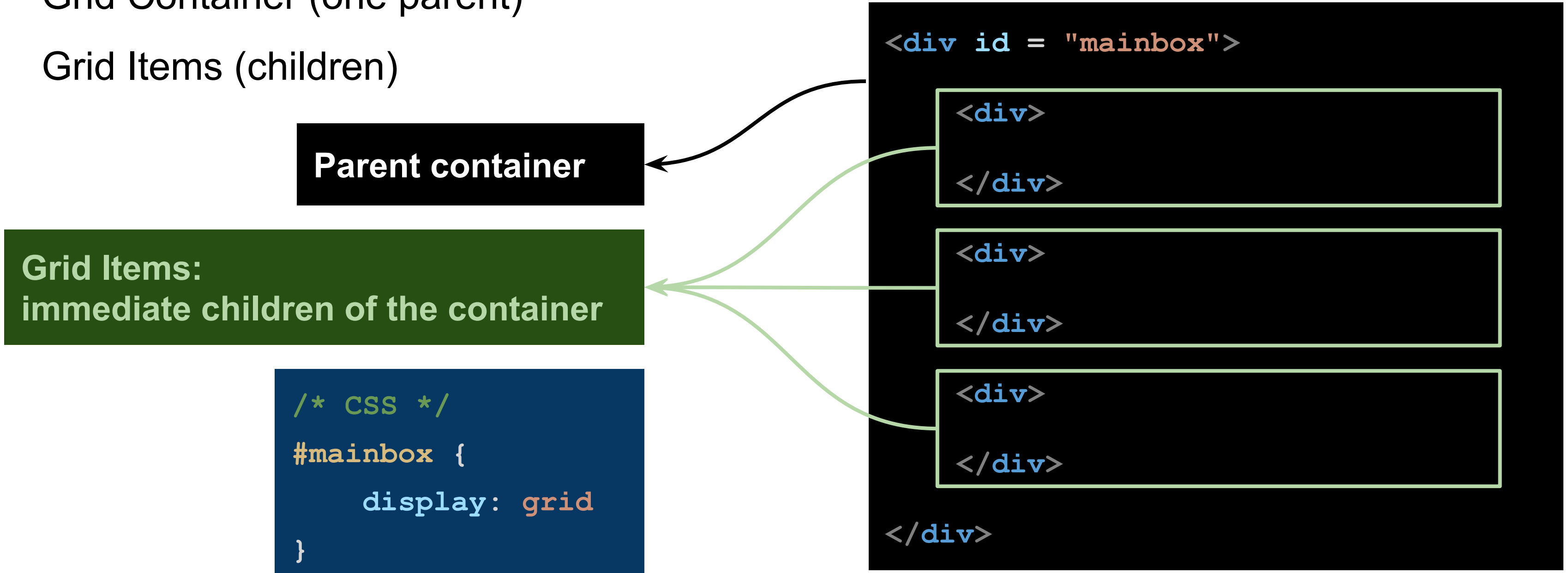
- Use CSS Grid to organize 2D layout of major elements (“macro”)
- Use CSS Flexbox to organize contents within an element (“micro”)
- The scale of macro/micro is subjective
- Resources:
  - A Complete Guide to Grid
  - A Complete Guide to Flexbox

CSS Grid (2D)

Reference: A Complete Guide to Grid

Elements of CSS Grid

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



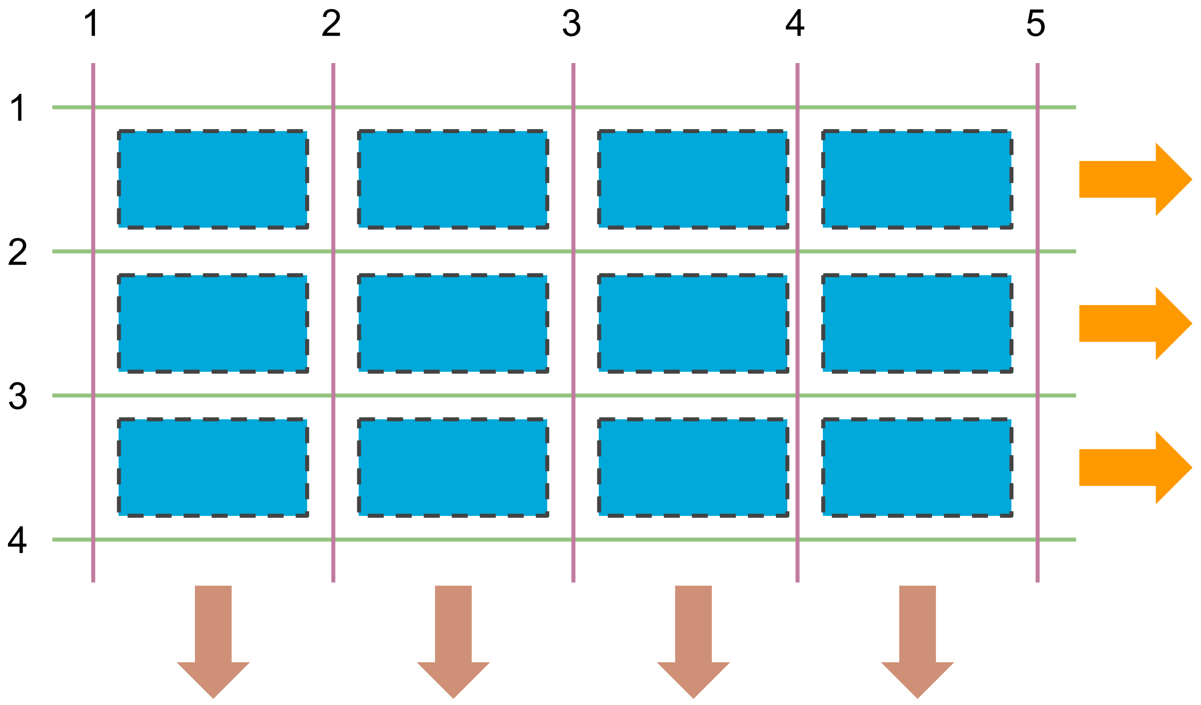
Parent container

Grid Items:
immediate children of the container

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

    </div>
    <div>

    </div>
    <div>

    </div>
</div>
```

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

Grid Details: Lines & Cells & Tracks



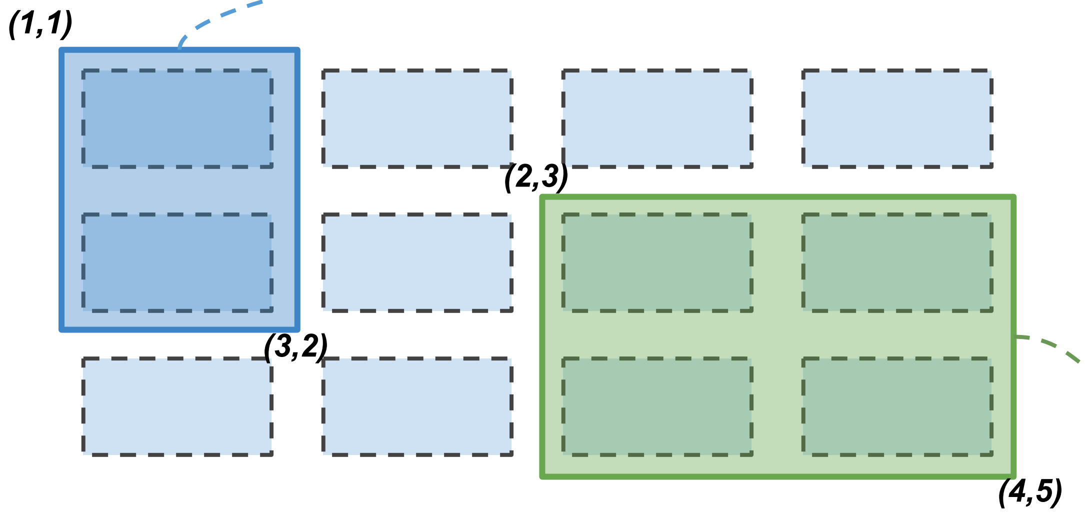
12 cells
5 vertical lines
4 horizontal lines

3 horizontal tracks

4 vertical tracks

Grid (Rectangular) Areas

Items may occupy multiple cells

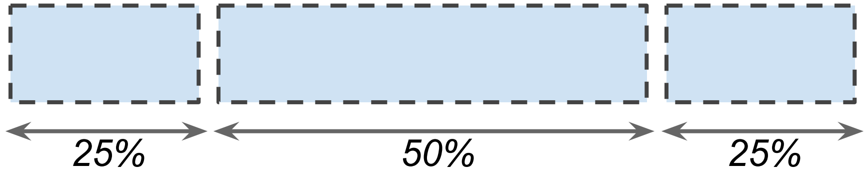


```
#blueBox {
    /* grid-row-start,
    grid-column-start,
    grid-row-end,
    and grid-column-end */
    grid-area: 1 / 1 / 3 / 2;
    background:blue;
}
```

```
#greenCorner {
    /* grid-row-start,
    grid-column-start,
    grid-row-end,
    and grid-column-end */
    grid-area: 2 / 3 / 4 / 5;
    background:green;
}
```

Grid Template (Rows|Columns)

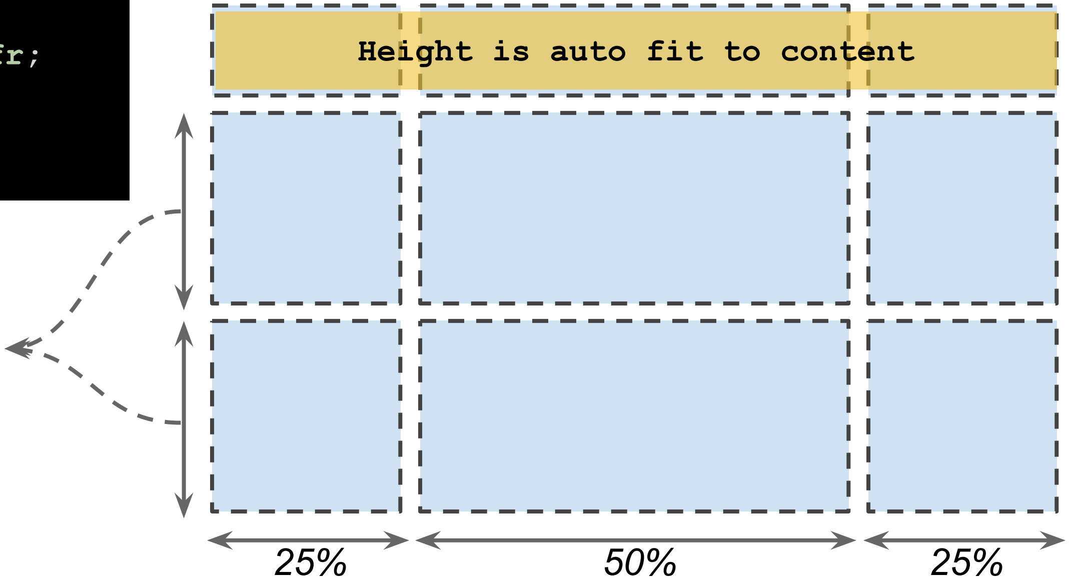
```
/* in CSS  */
#mainbox {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
}
```



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, …`	Fixed

Grid Template

```
/* in CSS  */
#mainbox {
    display: grid;
    gap: 5px;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: auto 1fr 1fr;
}
```



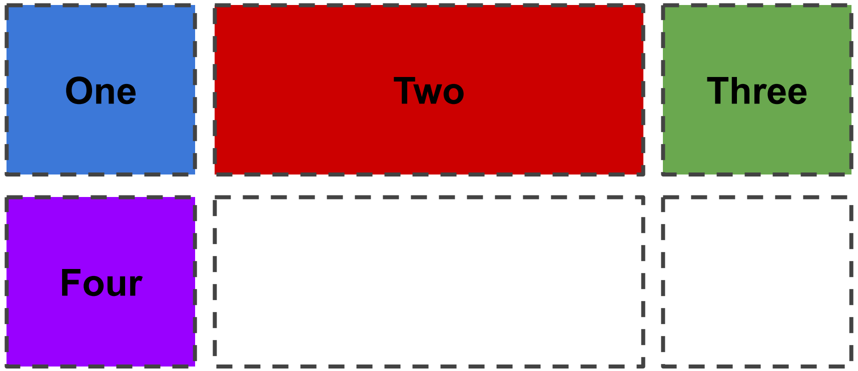
50% of **remaining** height each

Default Placement

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

```
/* in CSS  */
#mybox {
    display: grid;
    gap: 5px;
    grid-template-columns: 1fr 2fr 1fr;
}
```

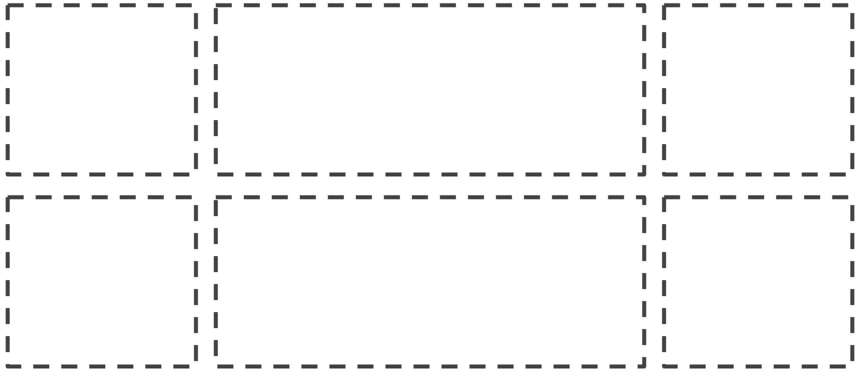
```
/* in HTML  */
<div id=”mybox”>
    <span class=”blue”>One</span>
    <span class=”red”>Two</span>
    <span class=”green”>Three</span>
    <span class=”purple”>Four</span>
</div>
```



Explicit positioning by "coordinates"

```
/* in CSS  */
#mybox {
    display: grid;
    gap: 5px;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 1fr 1fr;
}
.red {
    grid-row-start: 2;
    grid-row-end: 3;
    grid-column-start: 2;
    grid-column-end: 3;
}
```

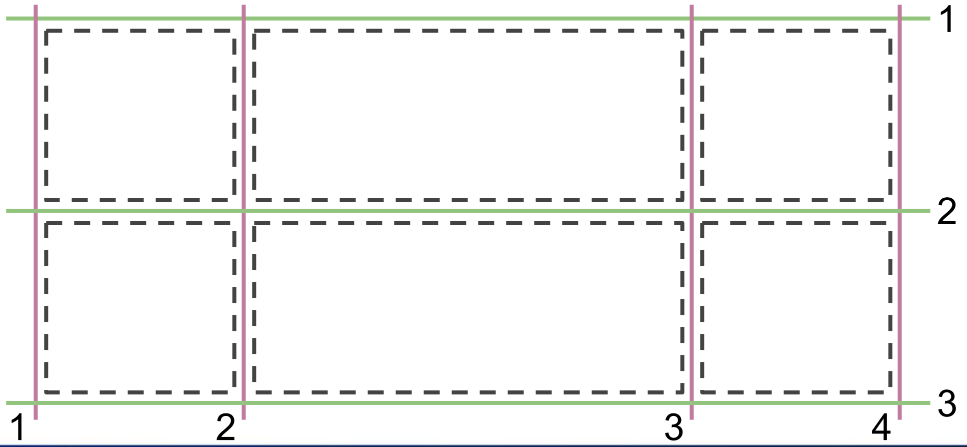
```
/* in HTML  */
<div id="mybox">
    <span class="blue">One</span>
    <span class="red">Two</span>
    <span class="green">Three</span>
    <span class="purple">Four</span>
</div>
```



Explicit positioning by “coordinates”

```
/* in CSS  */
#mybox {
    display: grid;
    gap: 5px;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 1fr 1fr;
}
.red {
    grid-row-start: 2;
    grid-row-end: 3;
    grid-column-start: 2;
    grid-column-end: 3;
}
```

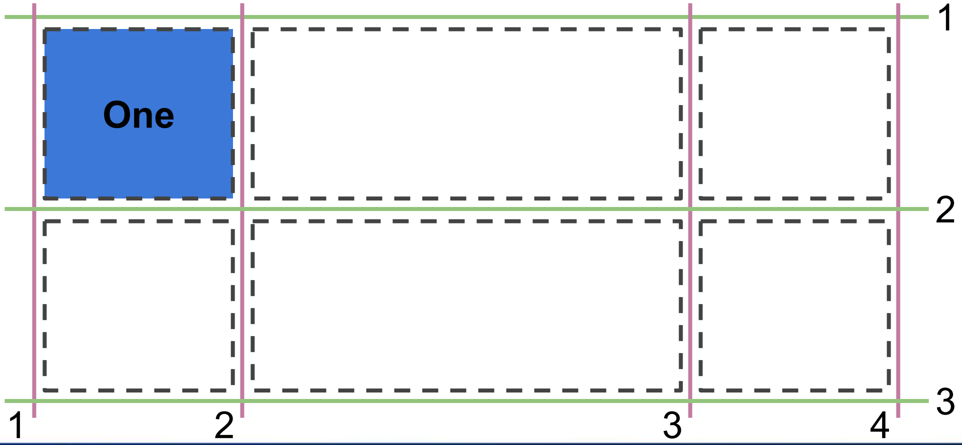
```
/* in HTML  */
<div id="mybox">
    <span class="blue">One</span>
    <span class="red">Two</span>
    <span class="green">Three</span>
    <span class="purple">Four</span>
</div>
```



Explicit positioning by "coordinates"

```
/* in CSS  */
#mybox {
    display: grid;
    gap: 5px;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 1fr 1fr;
}
.red {
    grid-row-start: 2;
    grid-row-end: 3;
    grid-column-start: 2;
    grid-column-end: 3;
}
```

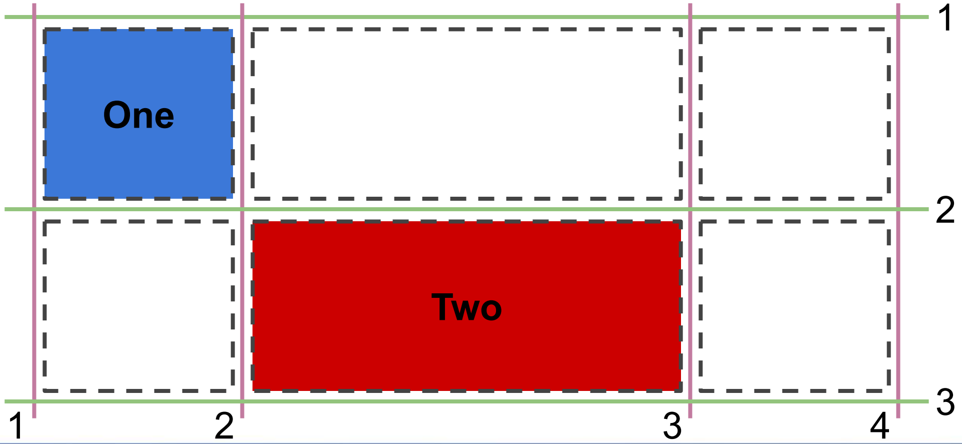
```
/* in HTML  */
<div id="mybox">
    <span class="blue">One</span>
    <span class="red">Two</span>
    <span class="green">Three</span>
    <span class="purple">Four</span>
</div>
```



Explicit positioning by “coordinates”

```
/* in CSS  */
#mybox {
    display: grid;
    gap: 5px;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 1fr 1fr;
}
.red {
    grid-row-start: 2;
    grid-column-start: 2;
    grid-column-end: 3;
}
```

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

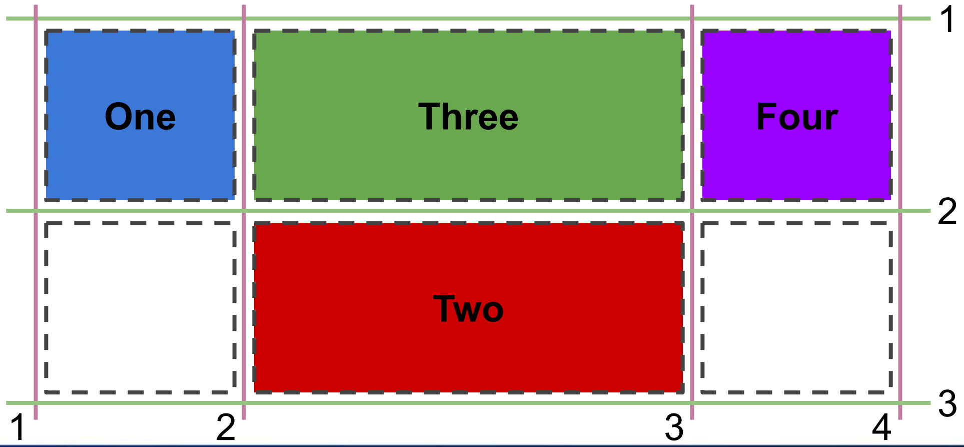


Explicit positioning by "coordinates"

```
/* in CSS  */
#mybox {
    display: grid;
    gap: 5px;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 1fr 1fr;
}
.red {
    grid-row-start: 2;
    grid-row-end: 3;
    grid-column-start: 2;
    grid-column-end: 3;
}
```

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

Default Placement

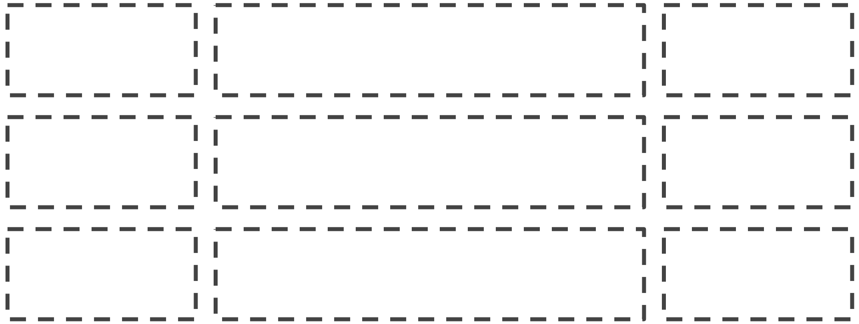


Explicit positioning by Area Names

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

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

```
.blue {
    background: blue;
    grid-area: logo
}

.red {
    background: red;
    grid-area: title
}
.green {
    background: green;
    grid-area: nav;
}
.purple {
    background: purple;
    grid-area: main;
}
```

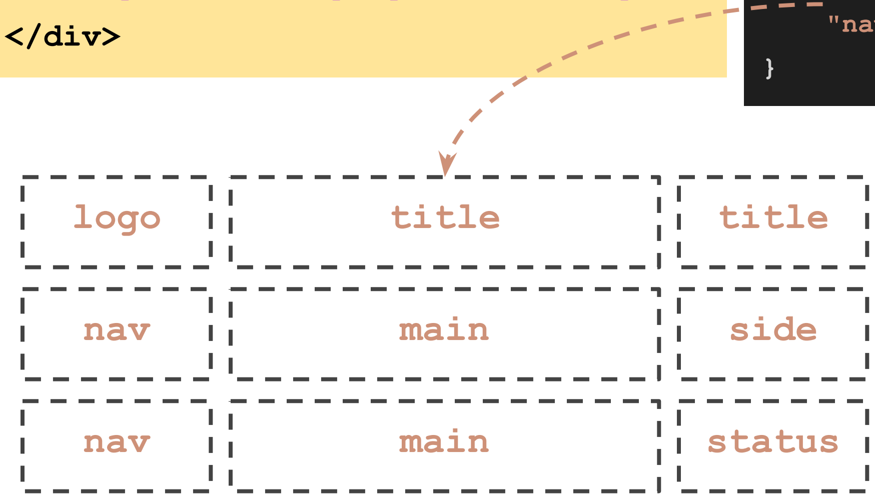


Explicit positioning by Area Names

```
/* in HTML  */
<div id="mybox">
     <span class="blue">Logo</span>
     <span class="red">Title</span>
     <span class="green">Nav</span>
     <span class="purple">Main</span>
</div>
```

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

```
.blue {
     background: blue;
     grid-area: logo
}

.red {
     background: red;
     grid-area: title
}
.green {
     background: green;
     grid-area: nav;
}
.purple {
     background: purple;
     grid-area: main;
}
```

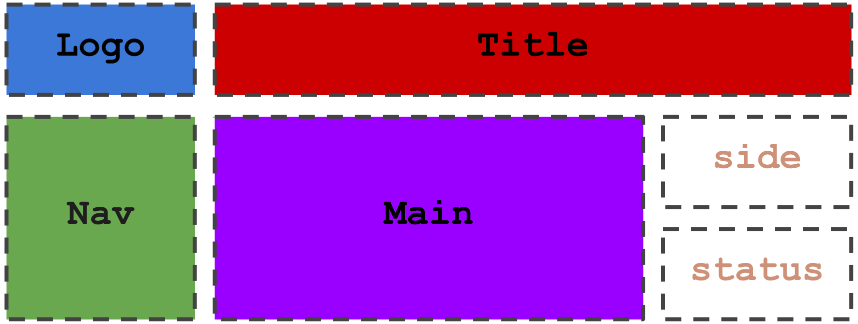


Explicit positioning by Area Names

```
/* in HTML  */
<div id="mybox">
    <span class="blue">Logo</span>
    <span class="red">Title</span>
    <span class="green">Nav</span>
    <span class="purple">Main</span>
</div>
```

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

```
.blue {
    background: blue;
    grid-area: logo
}

.red {
    background: red;
    grid-area: title
}
.green {
    background: green;
    grid-area: nav;
}
.purple {
    background: purple;
    grid-area: main;
}
```

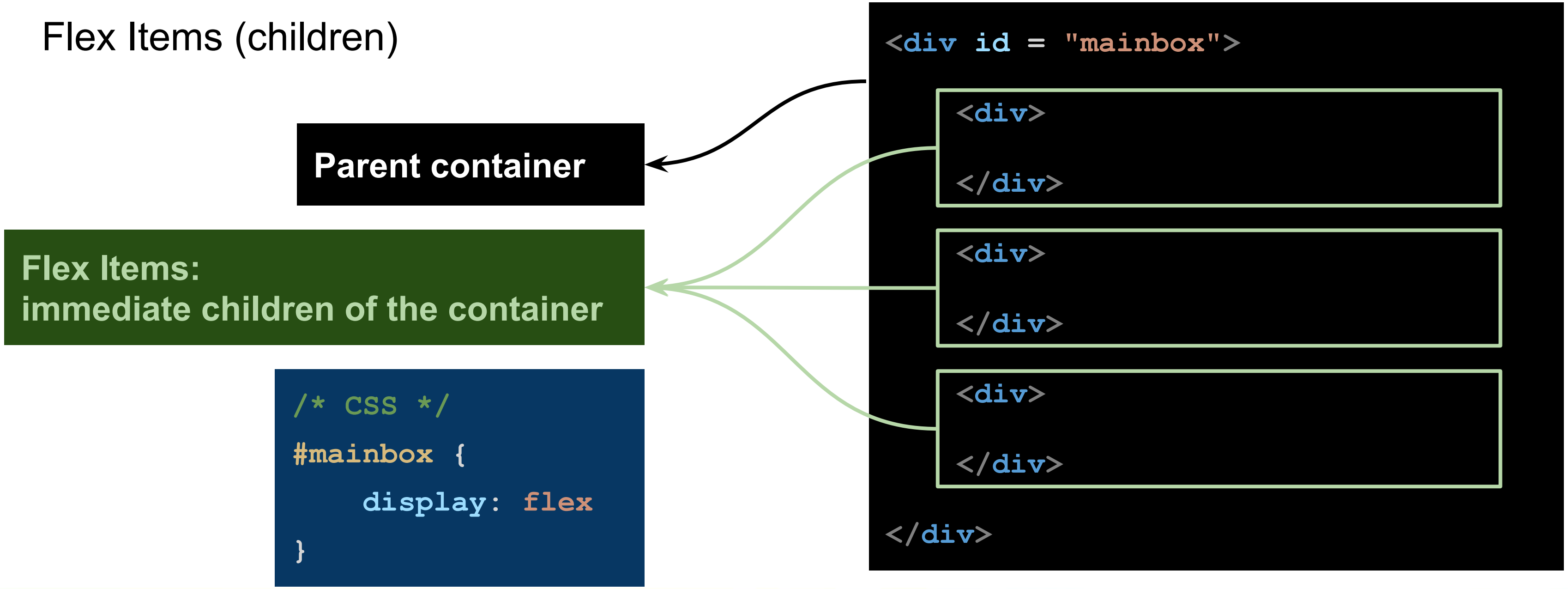


CSS Flexbox (1D)

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)



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

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

    </div>
    <div>

    </div>
    <div>

    </div>
</div>
```

Traditional Box vs. FlexBox

Traditional Box

Traditional layout “tricks”:

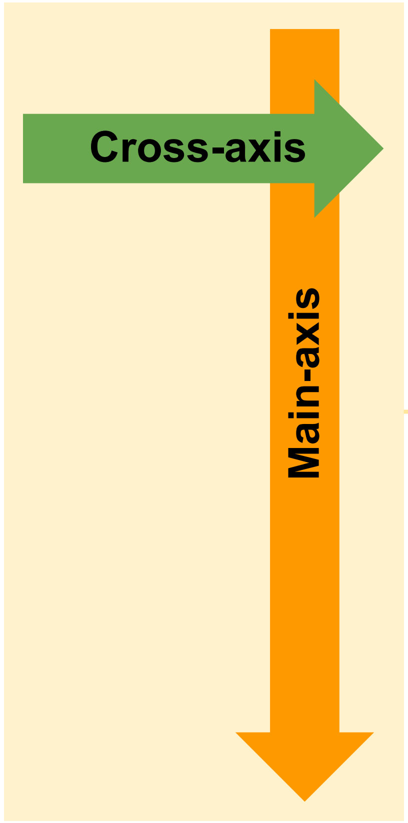
- display: (block|inline)
- float: (left|right)
- position: (fixed|absolute|relative)
- “grid” approach using <table>

FlexBox

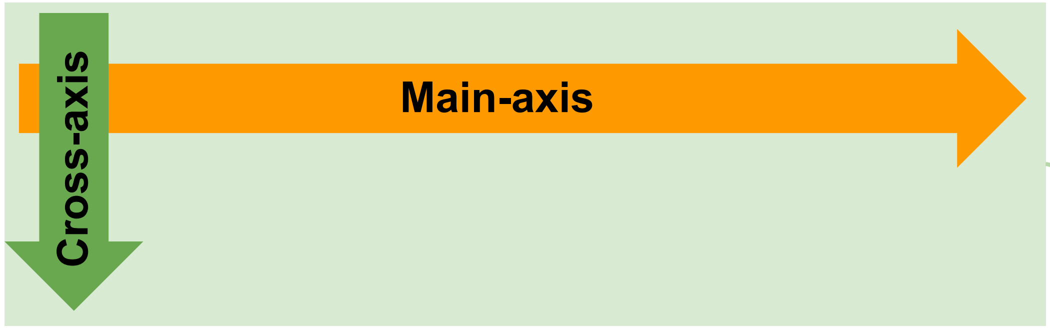
Modern approach

- **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;
}
```

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

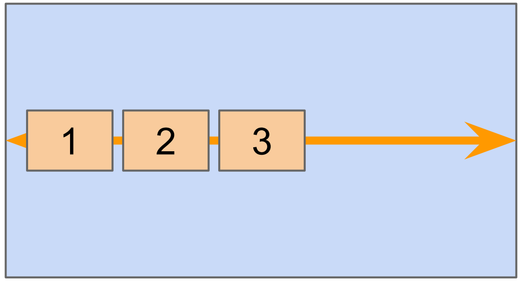
- order: override relative orders
- flex-grow: how items expand to fill up available extra space
- flex-shrink: disable/enable shrinking of elements when parent is shrunk
- align-self: override parent's align-items

Flexbox: Justification vs. Alignment

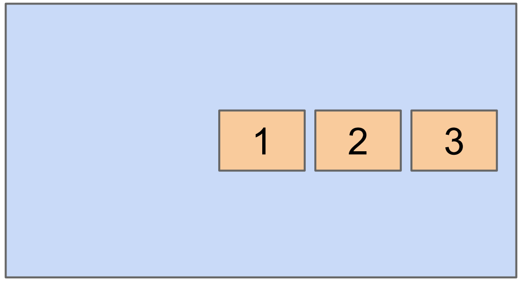
Property	Use by	Purpose
justify-content	parent	Placement of the entire contents along the major axis
align-items	parent	Placement of individual children along the minor axis
align-content	parent	Placement of the entire contents along the minor axis
align-self	child	Override the parent align-items property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

Justify-content (horizontal box)

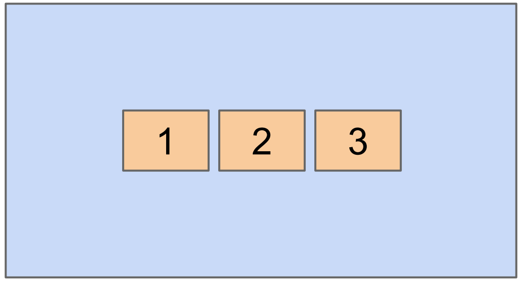
justify-content: flex-start



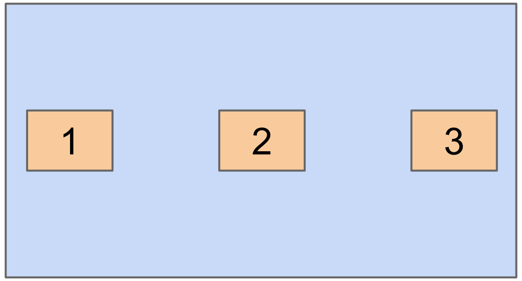
justify-content: flex-end



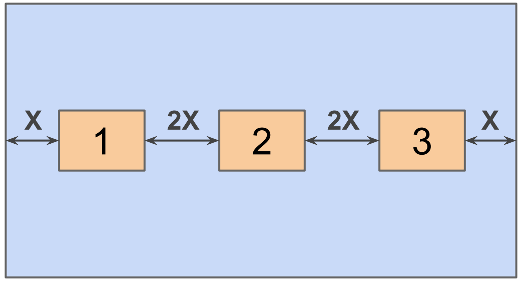
justify-content: center



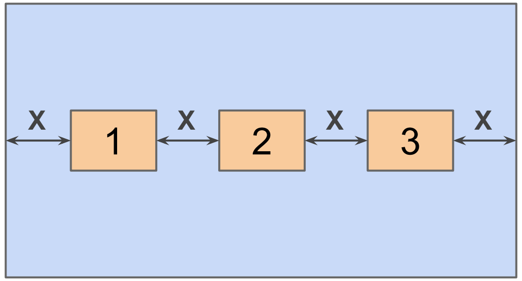
justify-content: space-between



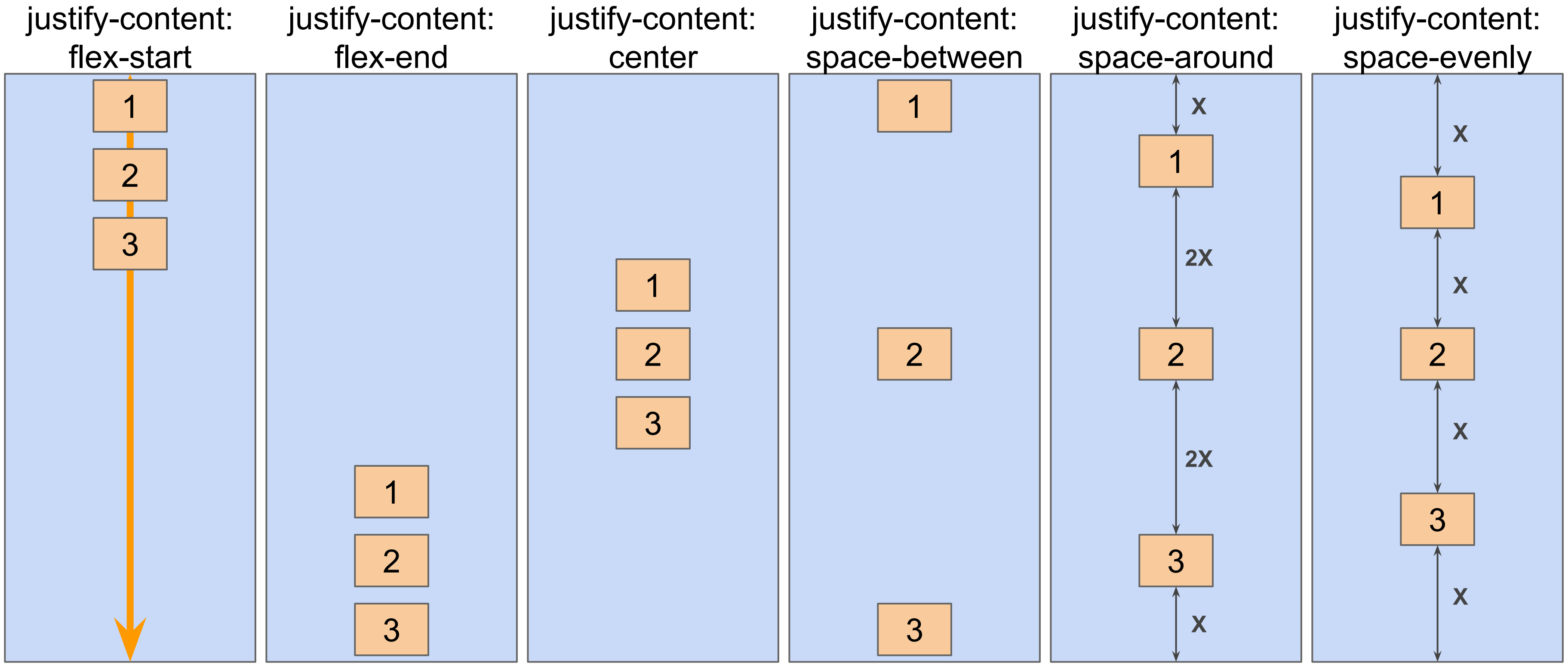
justify-content: space-around



justify-content: space-evenly

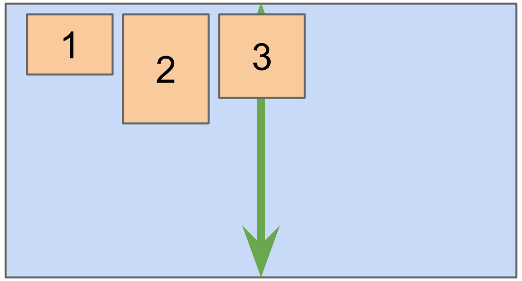


Justify-content (vertical box)

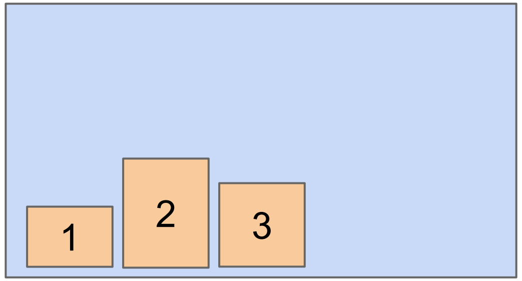

justify-content: flex-start
1
2
3
justify-content: flex-end
1
2
3
justify-content: center
1
2
3
justify-content: space-between
1
2
3
justify-content: space-around
X
1
2X
2
2X
3
X
justify-content: space-evenly
X
1
X
2
X
3
X

Align-items (horizontal box)

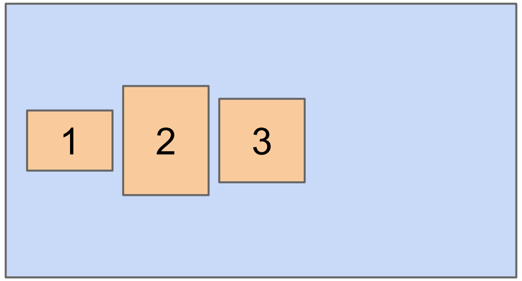
align-items: flex-start



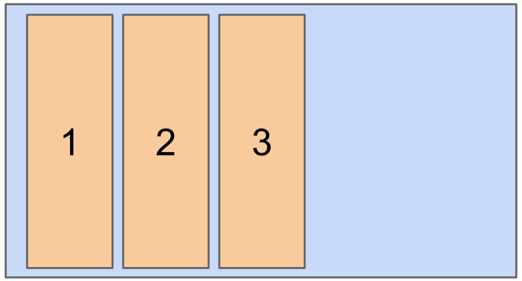
align-items: flex-end



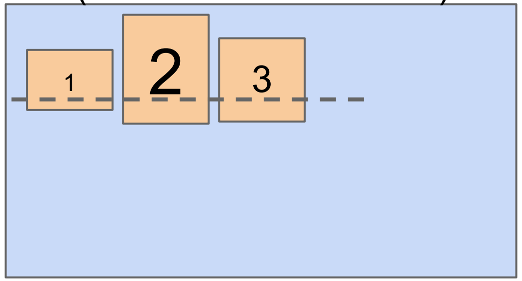
align-items: center



align-items: stretch

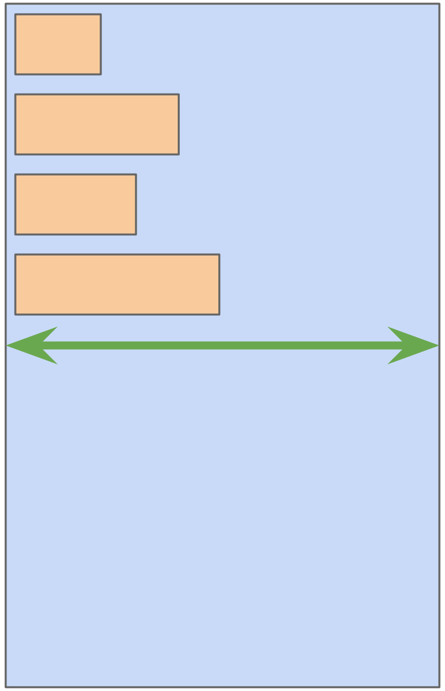


align-items: baseline
(the bottom of the text)

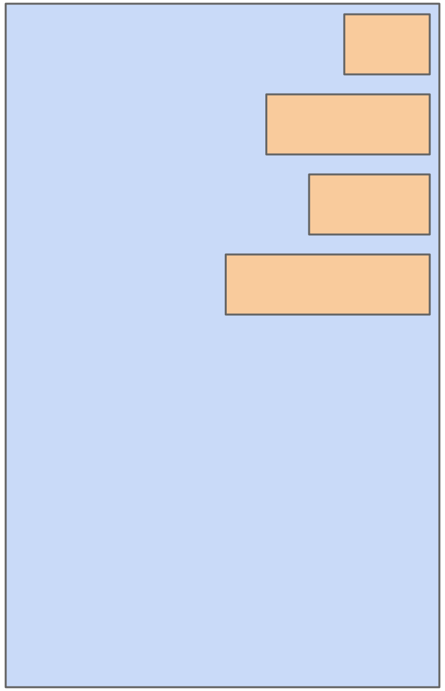


Align-items (vertical box)

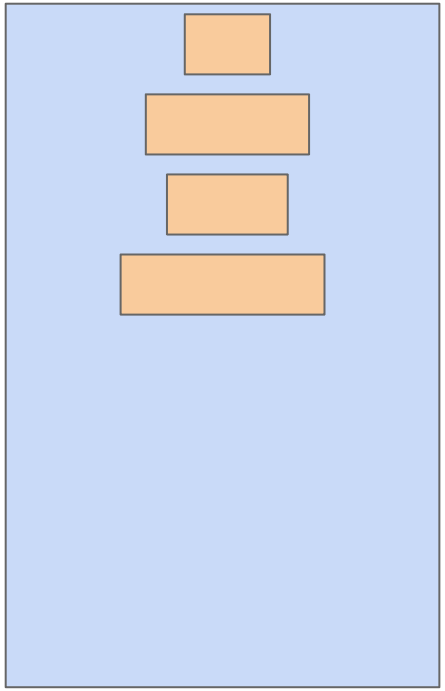
align-items: flex-start



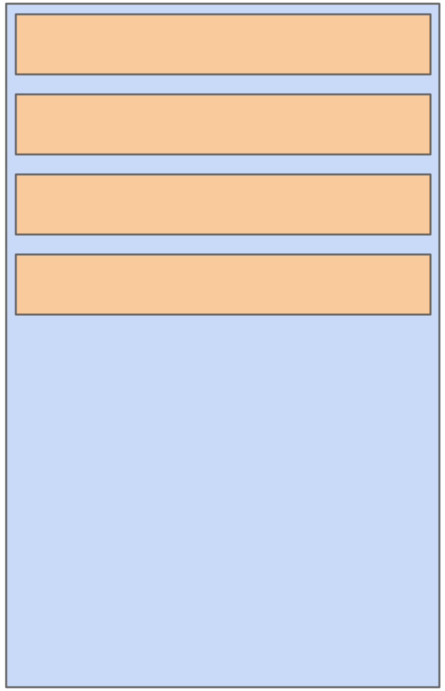
align-items: flex-end



align-items: center

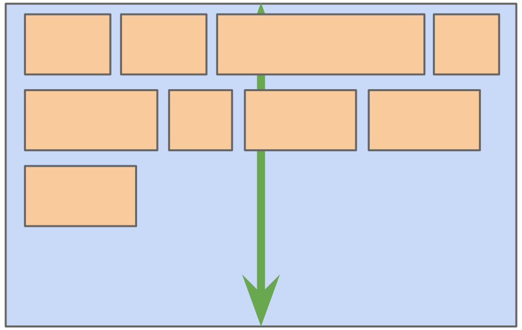


align-items: stretch

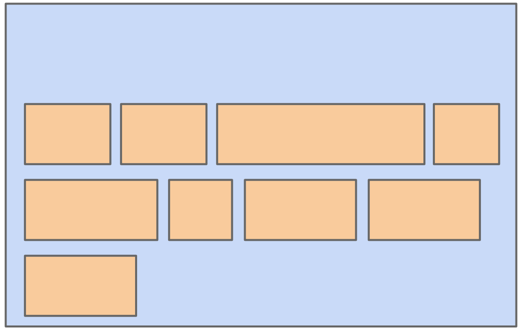


Align-content (horizontal box)

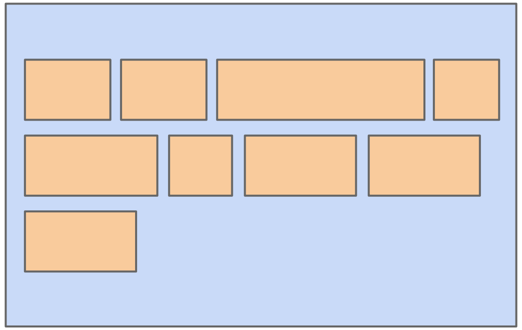
align-content: flex-start



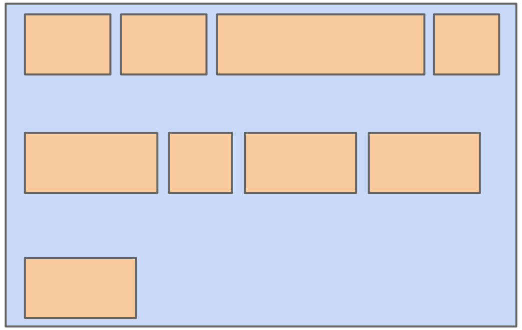
align-content: flex-end



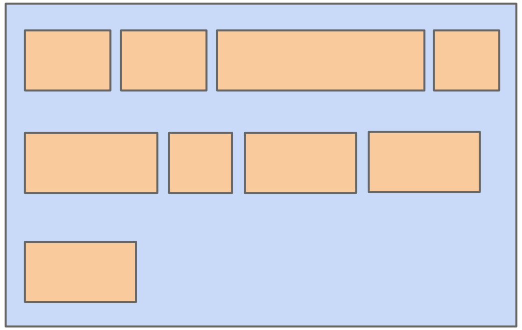
align-content: center



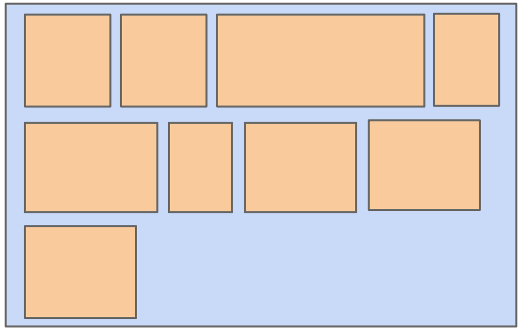
align-content: space-between



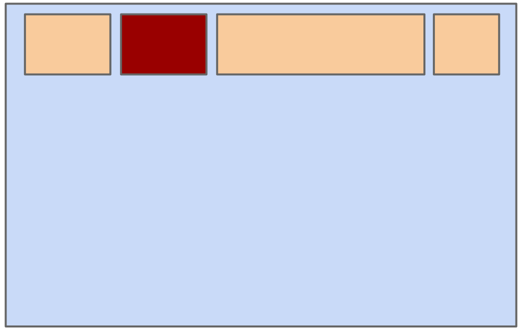
align-content: space-around



align-content: stretch

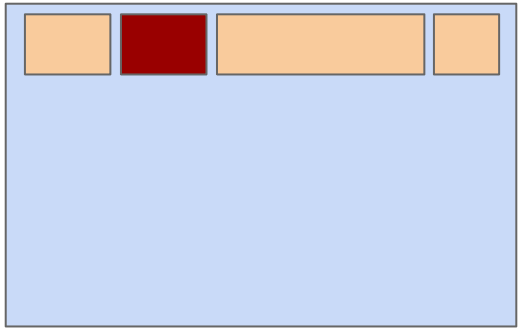


Align-self (horizontal box)



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

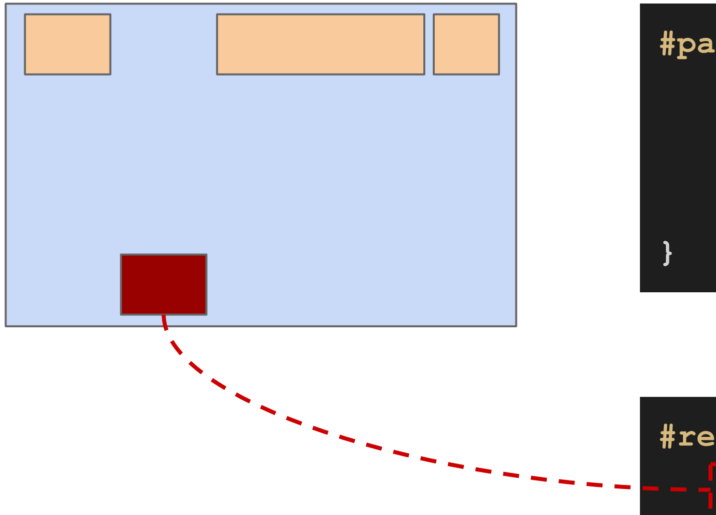
Align-self (horizontal box)



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

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

Align-self (horizontal box)



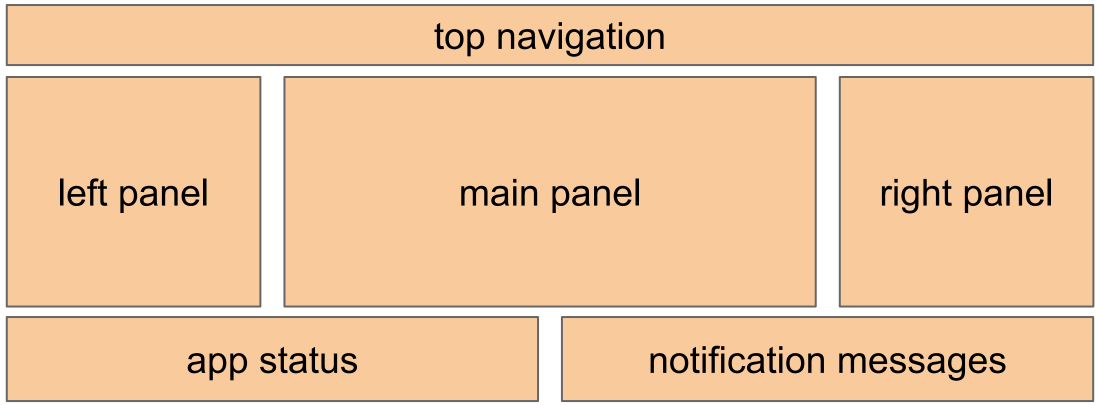
```
#parent-box {
    display: flex;
    flex-direction: row;
    align-items: flex-start;
}
```

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

Exercises

You UI designer team has decided to use the following layout for the front page of your web app

- The entire layout should fill up the entire browser canvas,
- The height of the top navigation is 40px,
- The height of the app status and notification messages is 60px,
- The left and right panel take 25% of the width each,
- The app status and notification message panel takes 50% of the width each



Design your HTML and CSS to implement this layout using CSS grid