Chapter 05

Introduction to **Web Design**

Layouts and Variables in CSS

Prepared by **Hok Tin**

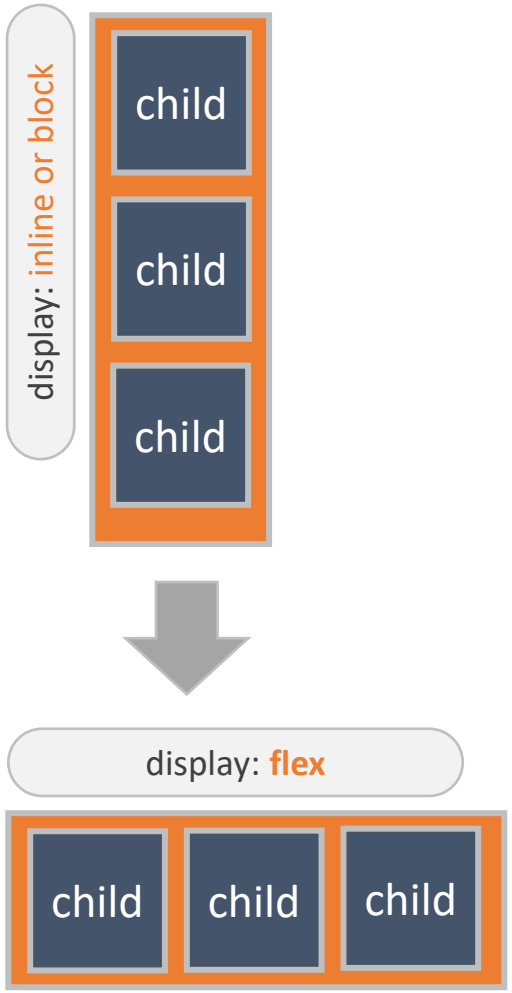
Content

- Flex layout
- Grid layout
- Row-column layout tips
- Variables

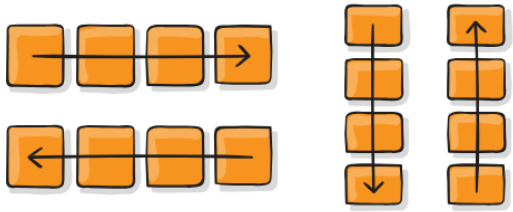
Flex layout

❑ Flexbox layout

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

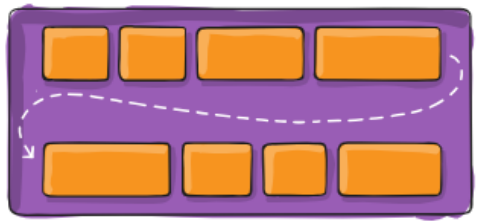


- **flex-direction**



```
.container {
  display: flex;
  flex-direction: row | row-reverse | column | column-reverse;
}
```

- **flex-wrap**



```
.container {
  display: flex;
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

Flex layout

❑ Flexbox layout

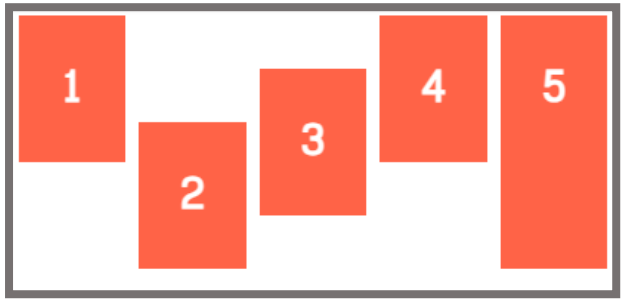
- **Flex-grow**

  This defines the ability for a flex item to grow if necessary


```
#child1 {flex-grow: 1;}
#child2 {flex-grow: 2;}
#child3 {flex-grow: 1;}
```

- **Align-self**

  Align itself in the flex container depending on the align-self value



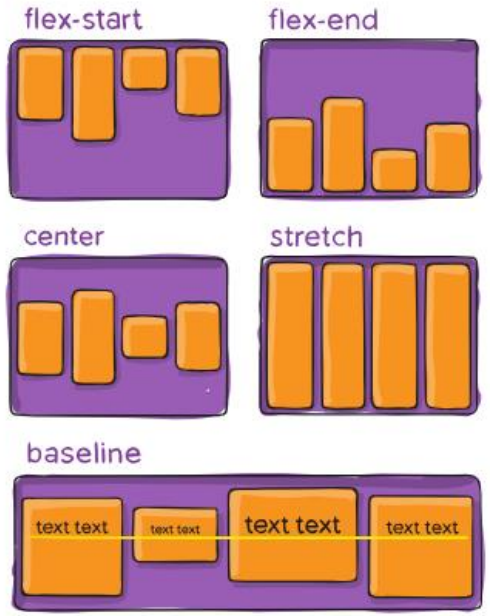

```
.child1 { align-self: flex-start; }
.child2 { align-self: flex-end; }
.child3 { align-self: center; }
.child4 { align-self: baseline; }
.child5 { align-self: stretch; }
```

Flex layout

❑ Flexbox layout

- **Align-items**

  This defines the default behavior for how flex items are laid out along the ***cross axis***

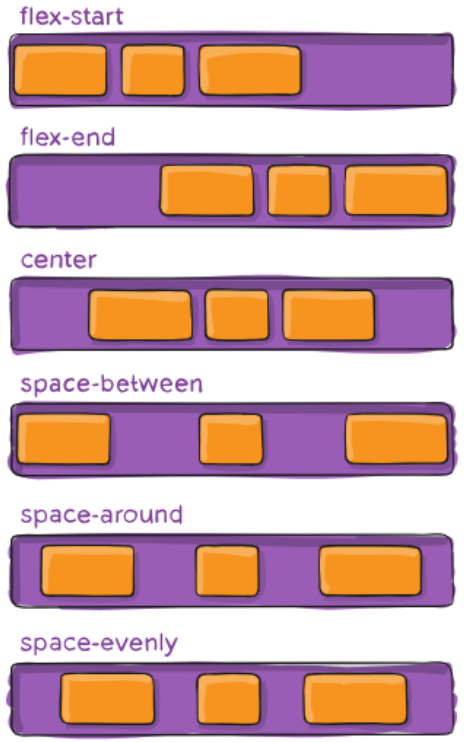



```
.container {
  align-item: flex-start | flex-end | center;
}
```

Others: baseline | first baseline | last baseline | start | end | self-start

- **Justify-content**

Alignment along the main ***axis***




```
.container {
  justify-content: flex-start | flex-end | center;
}
```

- **space-between**: items are evenly distributed in the line; first item is on the start line, last item on the end line
- **space-around**: items are evenly distributed in the line with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides.
- **space-evenly**: items are distributed so that the spacing between any two items
- Etc.

Grid layout

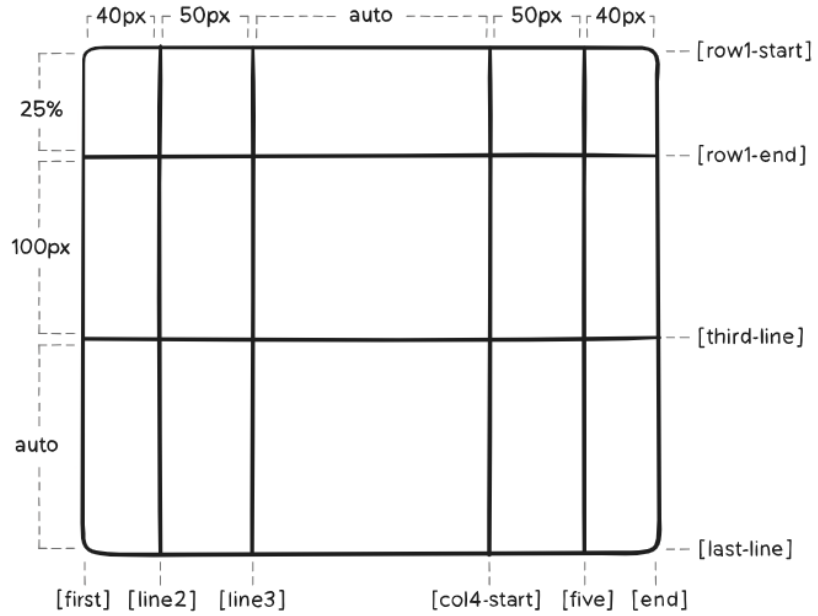
❑ Grid-template-columns && grid-template-rows

Defines the columns and rows of the grid with a space-separated list of values.

```
.container {
    grid-template-columns: ...  ...;
    /* e.g.
        1fr 1fr
        minmax(10px, 1fr) 3fr
        repeat(5, 1fr)
        50px auto 100px 1fr
    */
    grid-template-rows: ... ...;
    /* e.g.
        min-content 1fr min-content
        100px 1fr max-content
    */
}
```

- Set column and row values

```
.container {
  grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];
  grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];
}
```

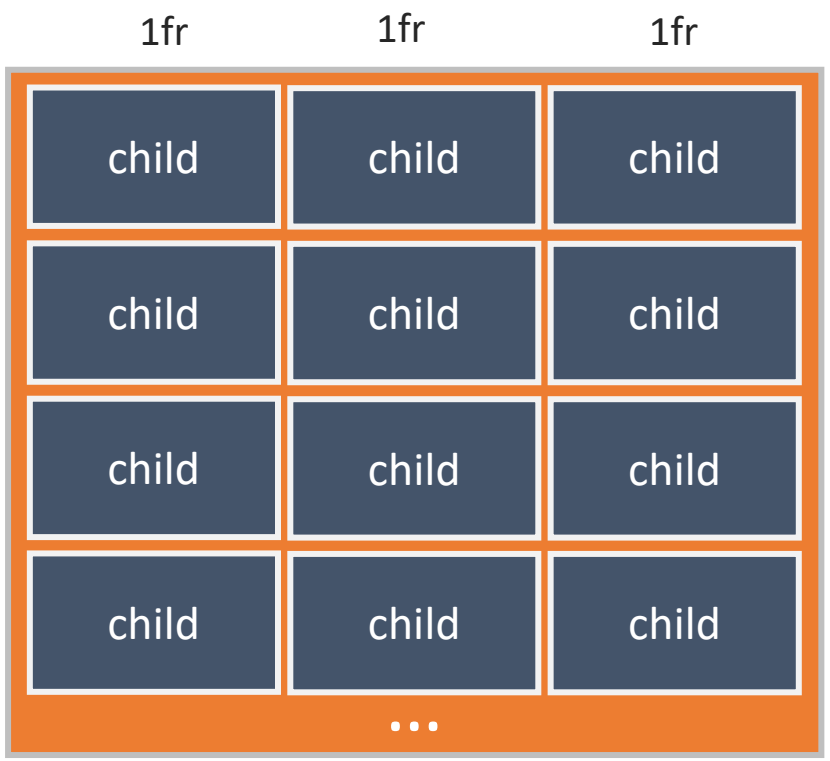


Grid layout

❑ Grid-template-columns && grid-template-rows

- Set auto column

```
.container {
    grid-template-columns: 1fr 1fr 1fr;
}
```

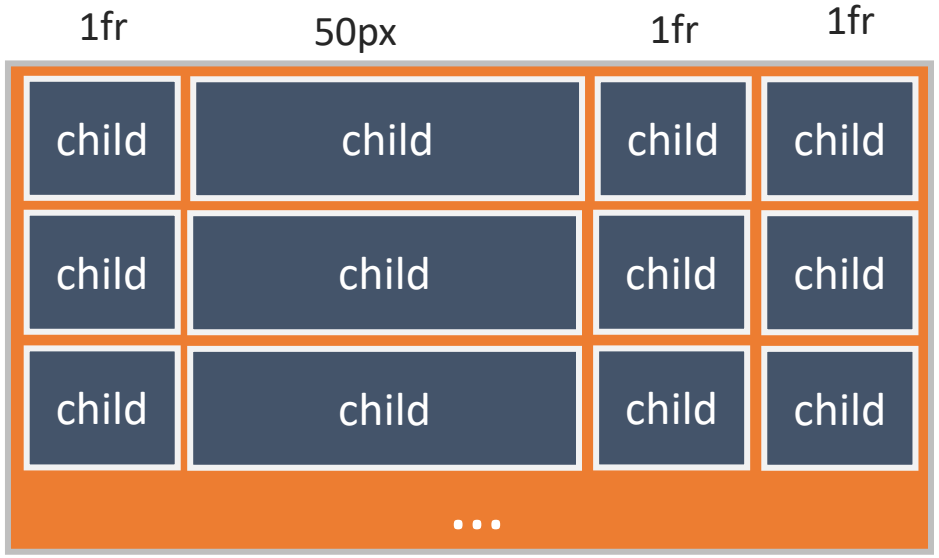


- Set specific number of "**Fr**" unit

```
.container {
  grid-template-columns: 1fr 3fr;
}
```

- Set auto and fixed-size column

```
.container {
    grid-template-columns: 1fr 50px 1fr 1fr;
}
```

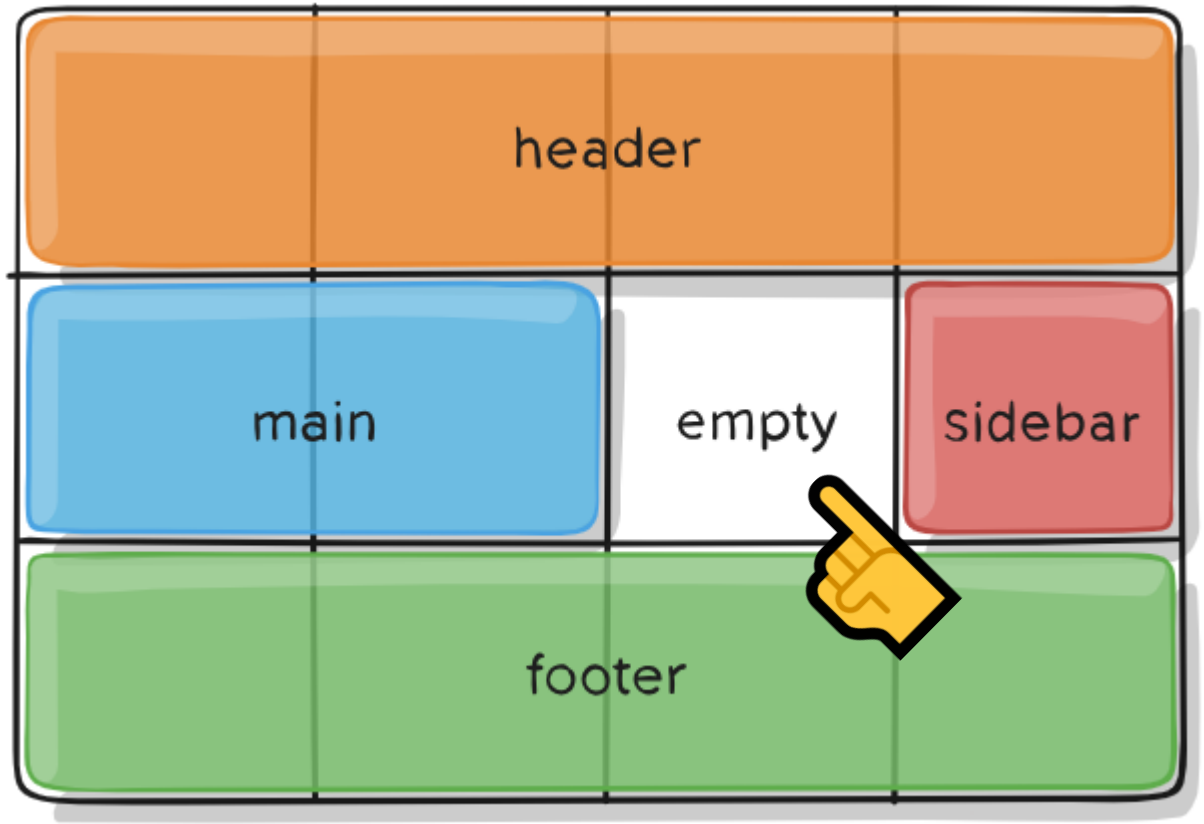


Grid layout

❑ Grid-template-areas

Defines a grid template by referencing the names of the grid areas which are specified with the grid-area property

```
.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
.item-c {
  grid-area: sidebar;
}
.item-d {
  grid-area: footer;
}

.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";
}
```

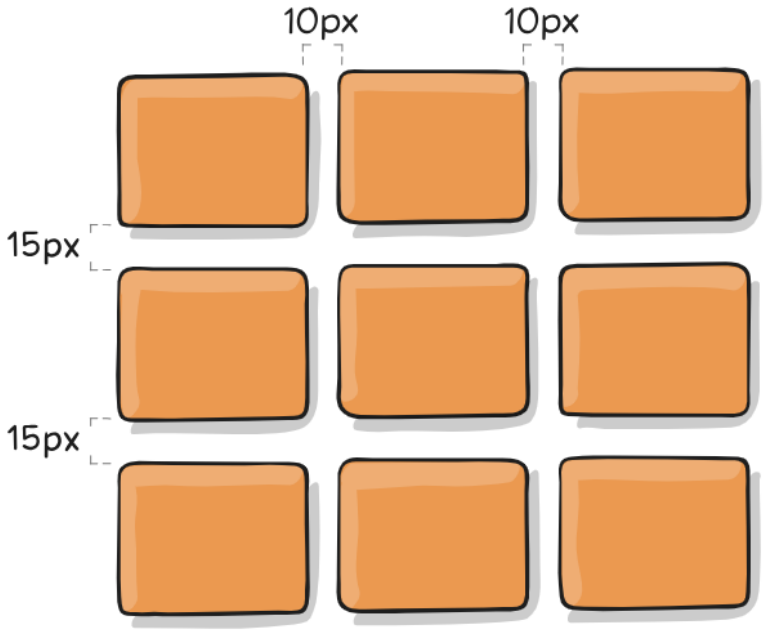


Grid layout

❑ Column-gap & row-gap

Specifies the size of the grid lines. You can think of it like setting the width of the gutters between the columns/rows

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

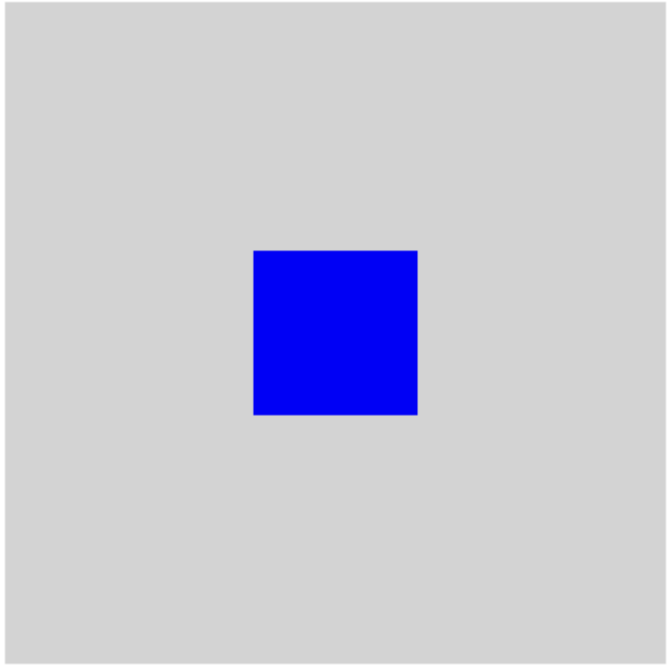


Row-column layout tips

❑ Centering element in flex

Align a child element to the position with flex

```
<!DOCTYPE html>
<html>
<head>
<style>
  .container {
    display: flex;
    justify-content: center; /* Horizontal centering */
    align-items: center; /* Vertical centering */
    background-color: lightgray;
    width: 200px;
    height: 200px;
  }

  .child {
    width: 50px;
    height: 50px;
    background-color: blue;
  }
</style>

</head>
<body>
  <div class="container">
    <div class="child"></div>
  </div>
</body>
</html>
```



- Justify-content: center | start | end
- Align-items: center | start | end

Row-column layout tips

❑ Row and column with Flex layout

- Using following classes to design your complex layout

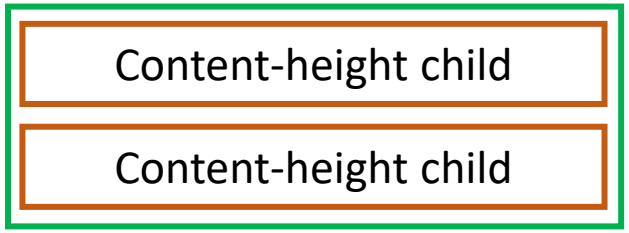
```css
.flex {
  display: flex;
}

.flex-row {
  flex-direction: row;
}

.flex-col {
  flex-direction: column;
}

.flex-row-0 {
  flex-grow: 0;
}

.flex-row-1 {
  flex-grow: 1;
}
```

- **Row**

```html
<div class="flex">
  <div class="flex-grow-0">Content-width child</div>
  <div class="flex-grow">Growing-width child</div>
</div>
```

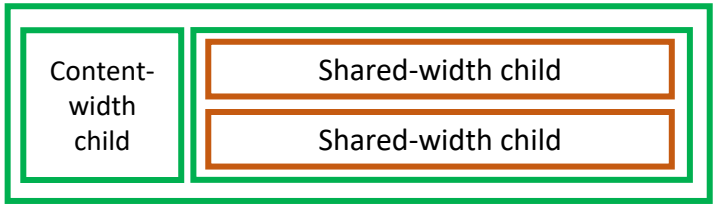
- **Column**

```html
<div class="flex flex-col">
  <div class="flex-grow-0">Content-height child</div>
  <div class="flex-grow-0">Content-height child</div>
</div>
```



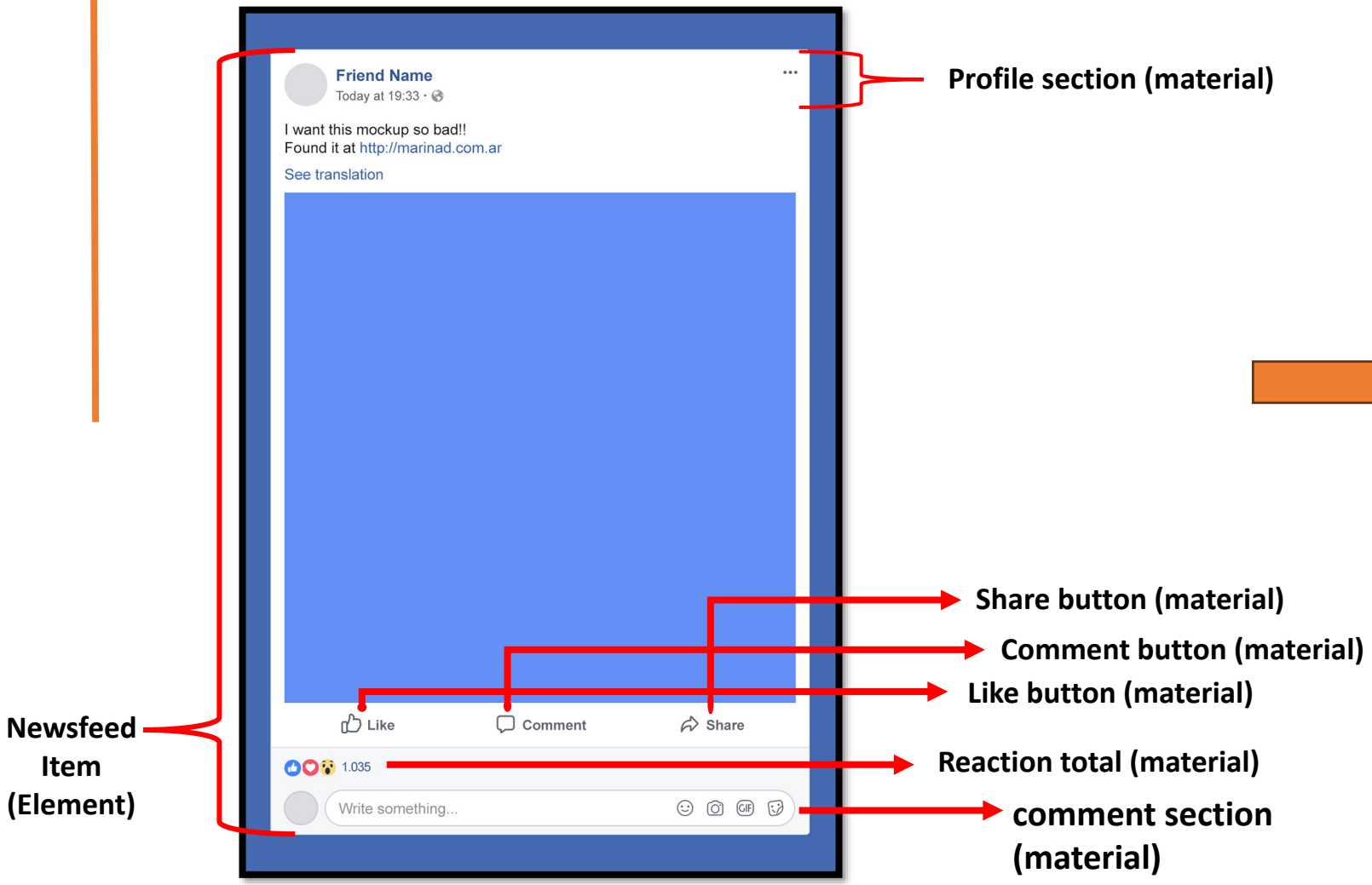
- **Nested row-column**

```html
<div class="flex">
  <div class="flex-grow-0">Content-width child</div>
  <div class="flex-grow-0 flex flex-col">
    <div class="flex-grow">Shared-width child</div>
    <div class="flex-grow">Shared-width child</div>
  </div>
</div>
```

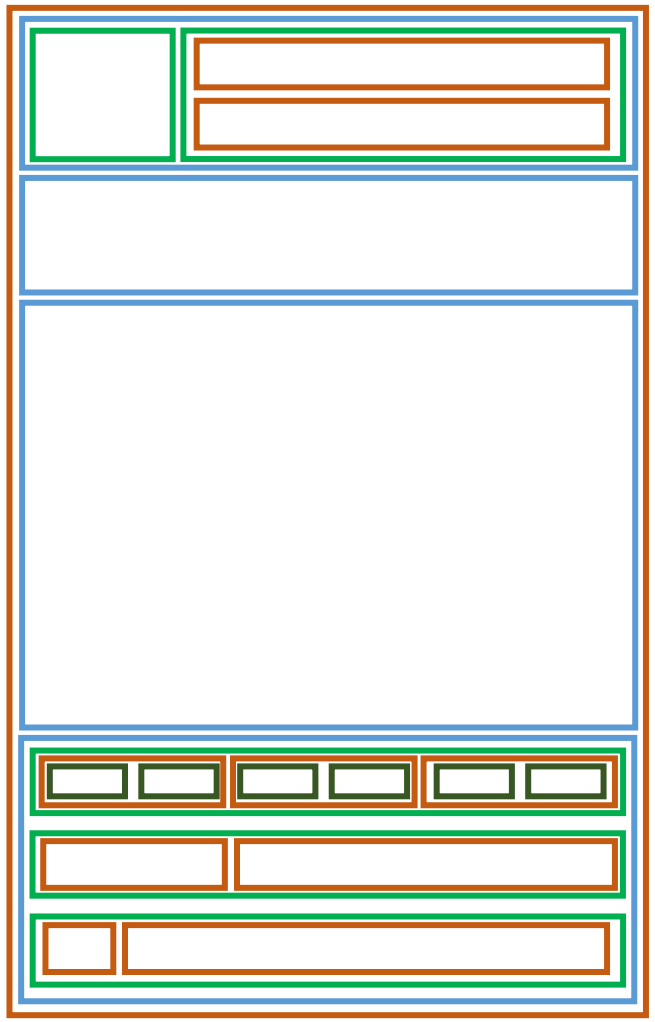


Row-column layout tips

- Let's design following Facebook Post card using your learnt tips

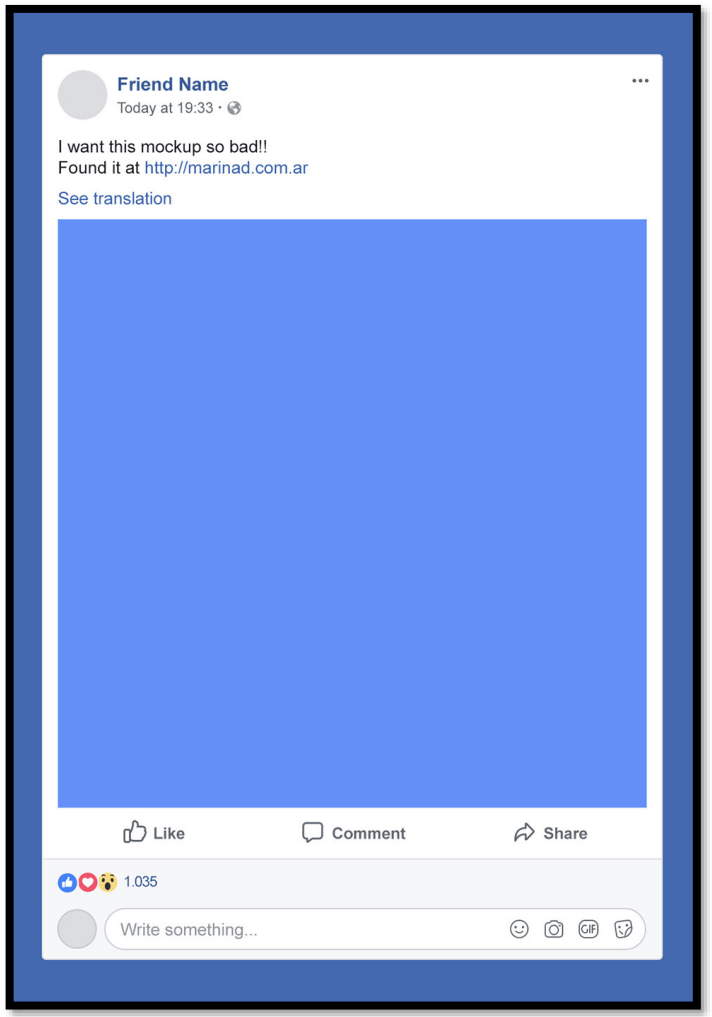


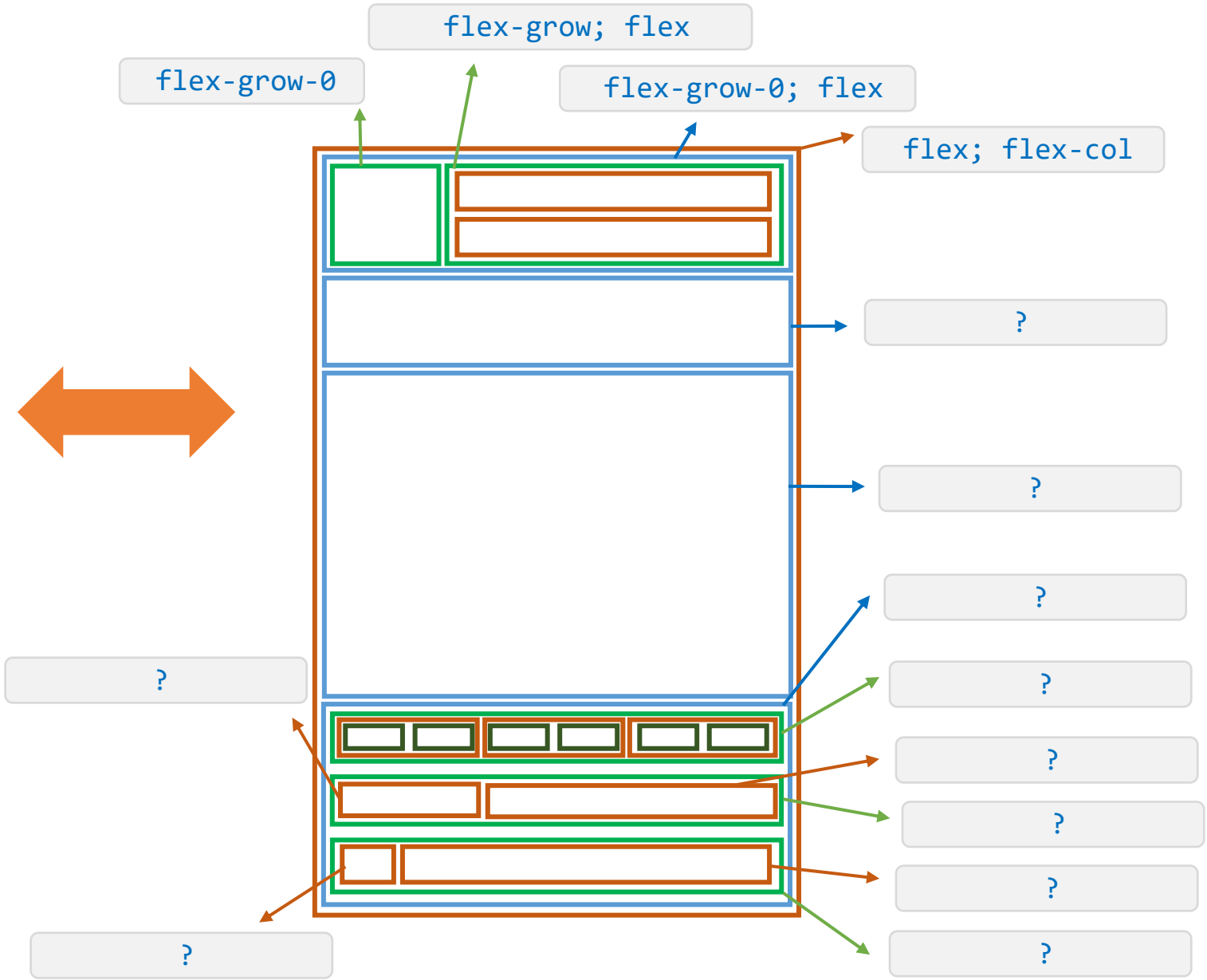
- You have to firstly define your row-column layout before doing something else



Row-column layout tips

- Row and column with Flex layout





Variables

❑ Variable declaration

- **Global variable**

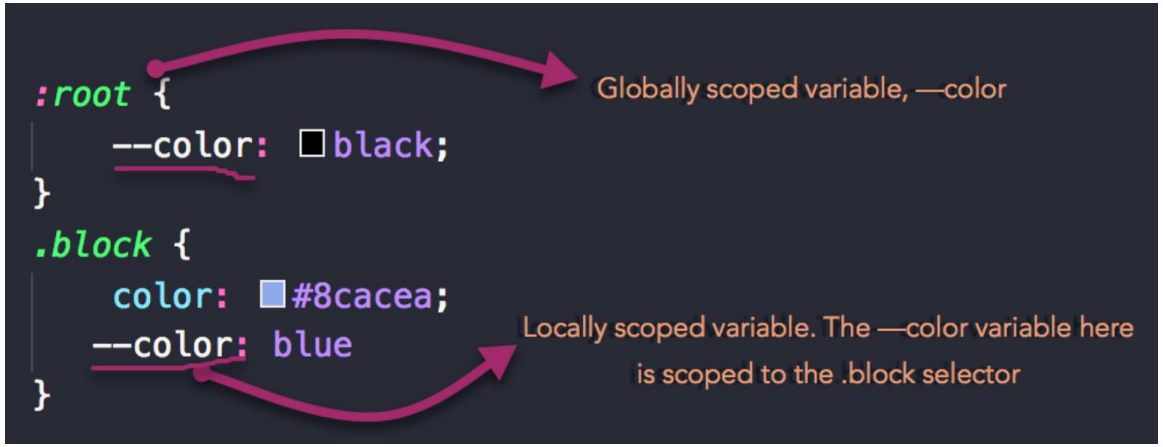
The :root selector allows you to target the highest-level element in the DOM, or document tree. So, variables declared in this way are kind of scoped to the global scope

```
:root {
  --main-color: red
}
```

- **Local variable**

Local CSS variables are scoped to the element where they are defined. In other words, they only apply to the elements with the same or nested selectors where the variable is defined.

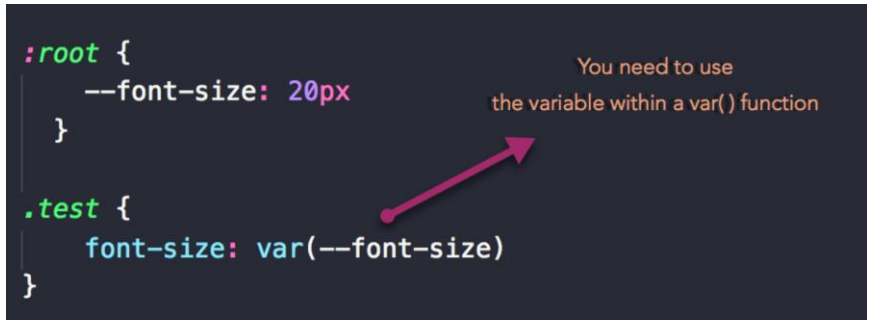
```
div {
  --color: red;
  color: var(--color);
}
```



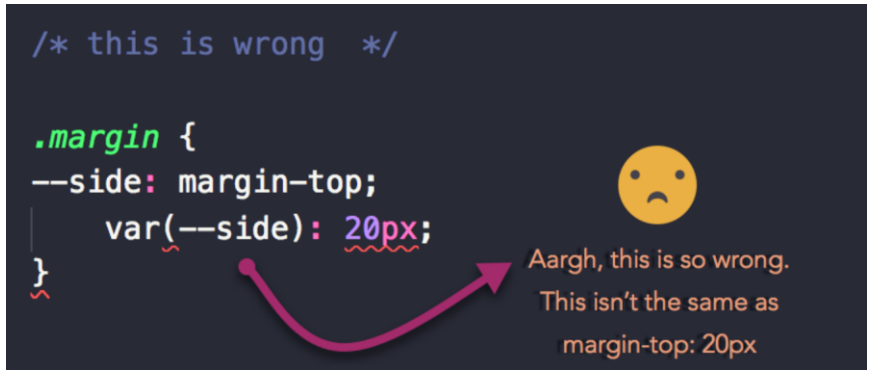
Variables

❑ Using CSS Variables

- The right way to use variable



- Wrong way to use variable



❖ **If you do math, then use the calc() function like so:**

- The right way to do math with variables

```
.margin {
  --space: calc(20px * 2);
  font-size:  var(--space);  /*equals 40px*/
}
```

- Wrong way to do math with variables

```
/*this is wrong */
.margin {
  --space: 20px * 2;
  font-size:  var(--space);  /*not 40px*/
}
```

Good luck 👌

References

1. https://www.educative.io/ (**The Complete Advanced Guide to CSS**)
2. https://css-tricks.com/snippets/css/a-guide-to-flexbox/
3. https://css-tricks.com/snippets/css/complete-guide-grid/
4. https://www.tutorialspoint.com/set-areas-within-the-grid-layout-in-css