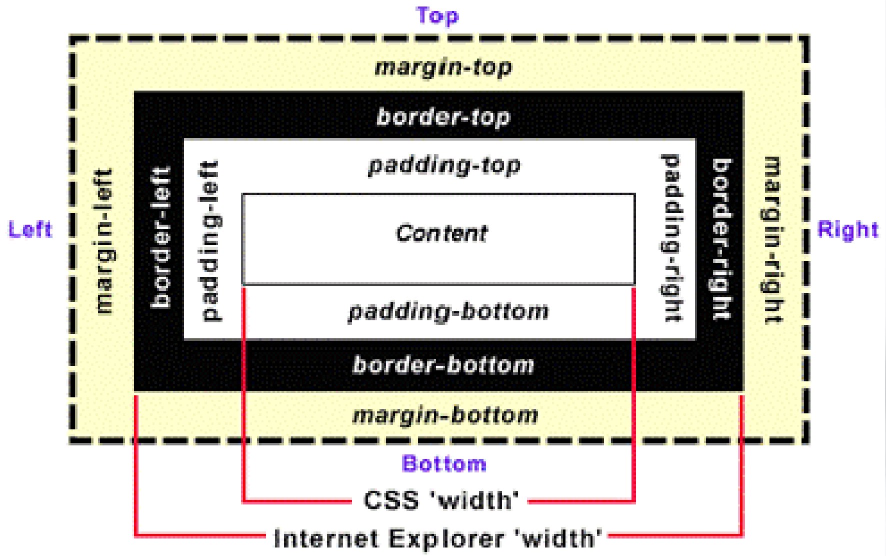
Front-end Development

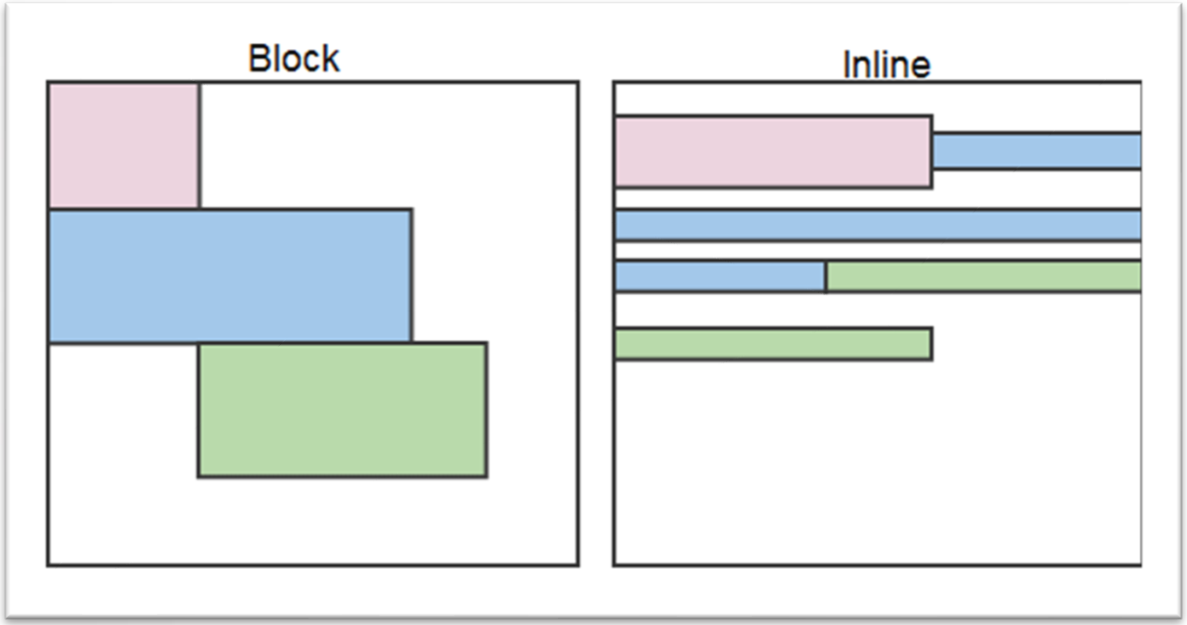
Lecturer: Ung Văn Giàu
Email: giau.ung@eiu.edu.vn



CSS Layout

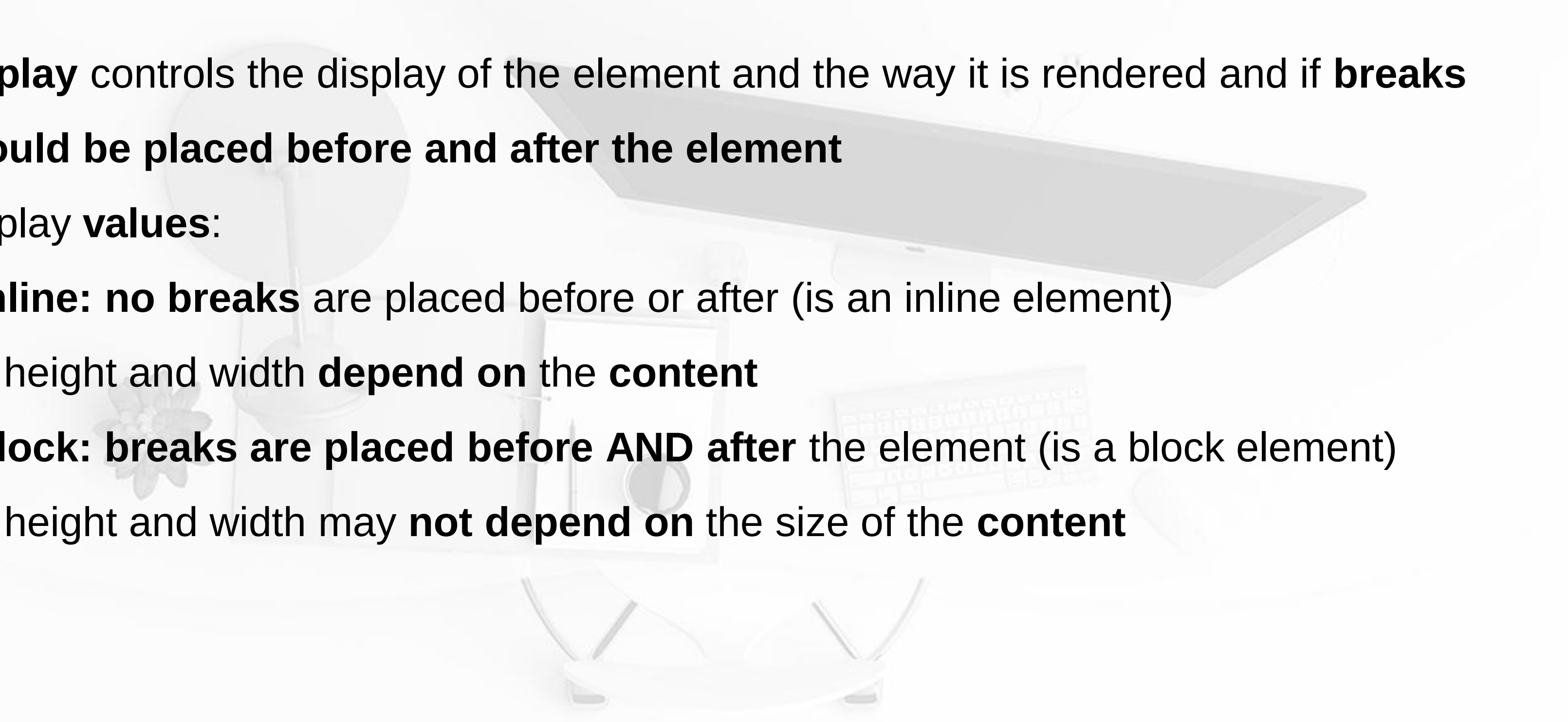
Control the arrangement of the HTML elements

Contents



1. Display

Display

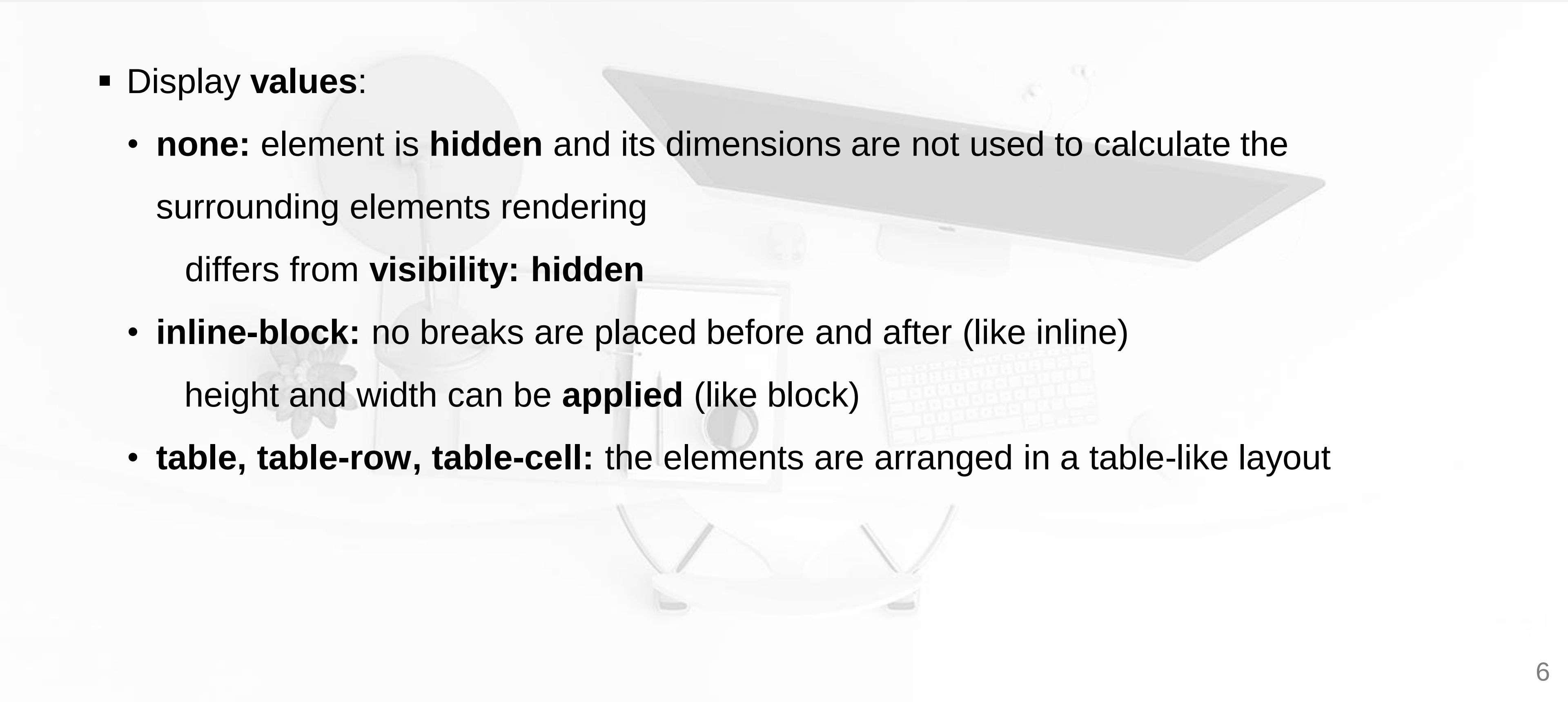


- **display** controls the display of the element and the way it is rendered and if **breaks should be placed before and after the element**
- Display **values**:
  - **inline: no breaks** are placed before or after (is an inline element)

    height and width **depend on** the **content**
  - **block: breaks are placed before AND after** the element (is a block element)

    height and width may **not depend on** the size of the **content**

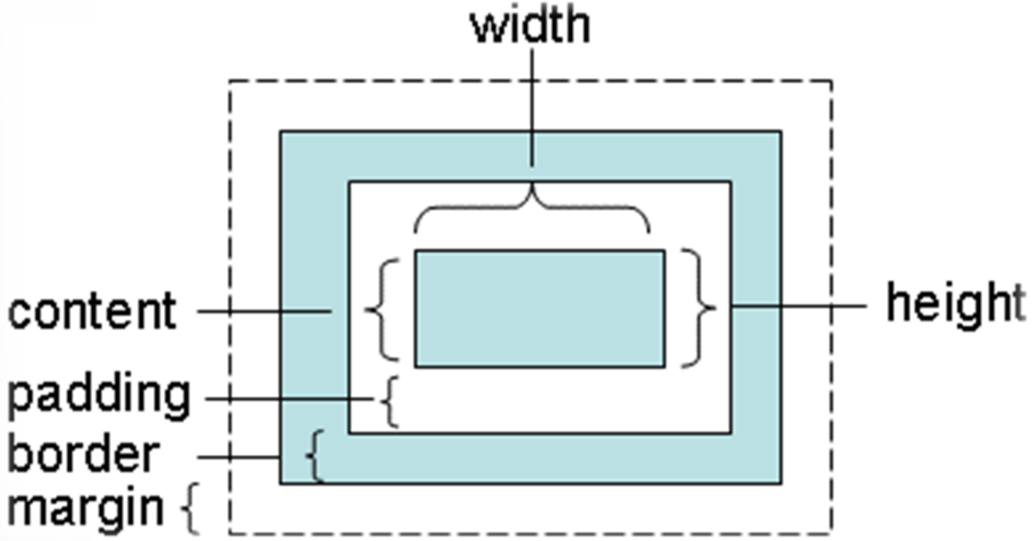
Display Values



- Display **values**:
  - **none:** element is **hidden** and its dimensions are not used to calculate the surrounding elements rendering

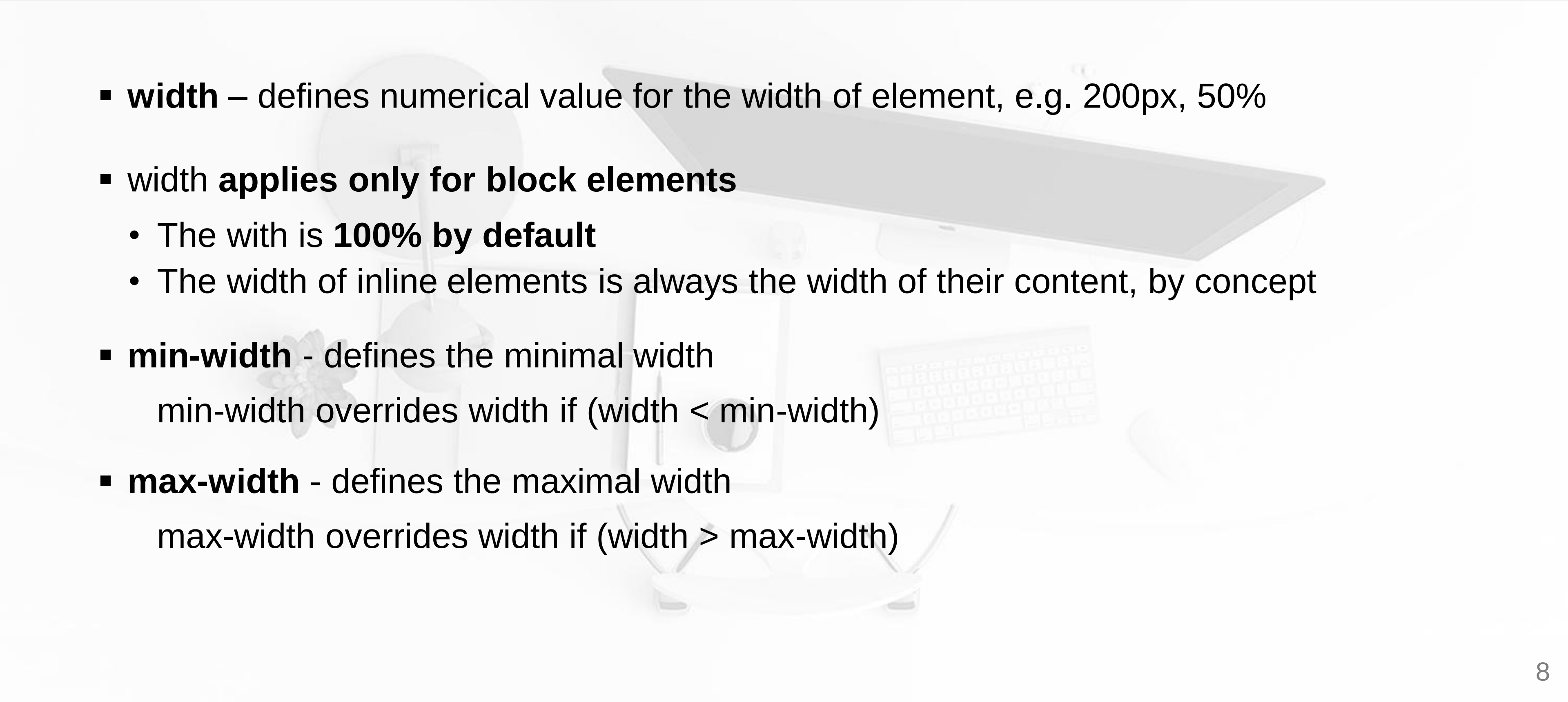
    differs from **visibility: hidden**
  - **inline-block:** no breaks are placed before and after (like inline)

    height and width can be **applied** (like block)
  - **table, table-row, table-cell:** the elements are arranged in a table-like layout



2. Width/Height

Width



- **width** – defines numerical value for the width of element, e.g. 200px, 50%

- width **applies only for block elements**
  - The with is **100% by default**
  - The width of inline elements is always the width of their content, by concept

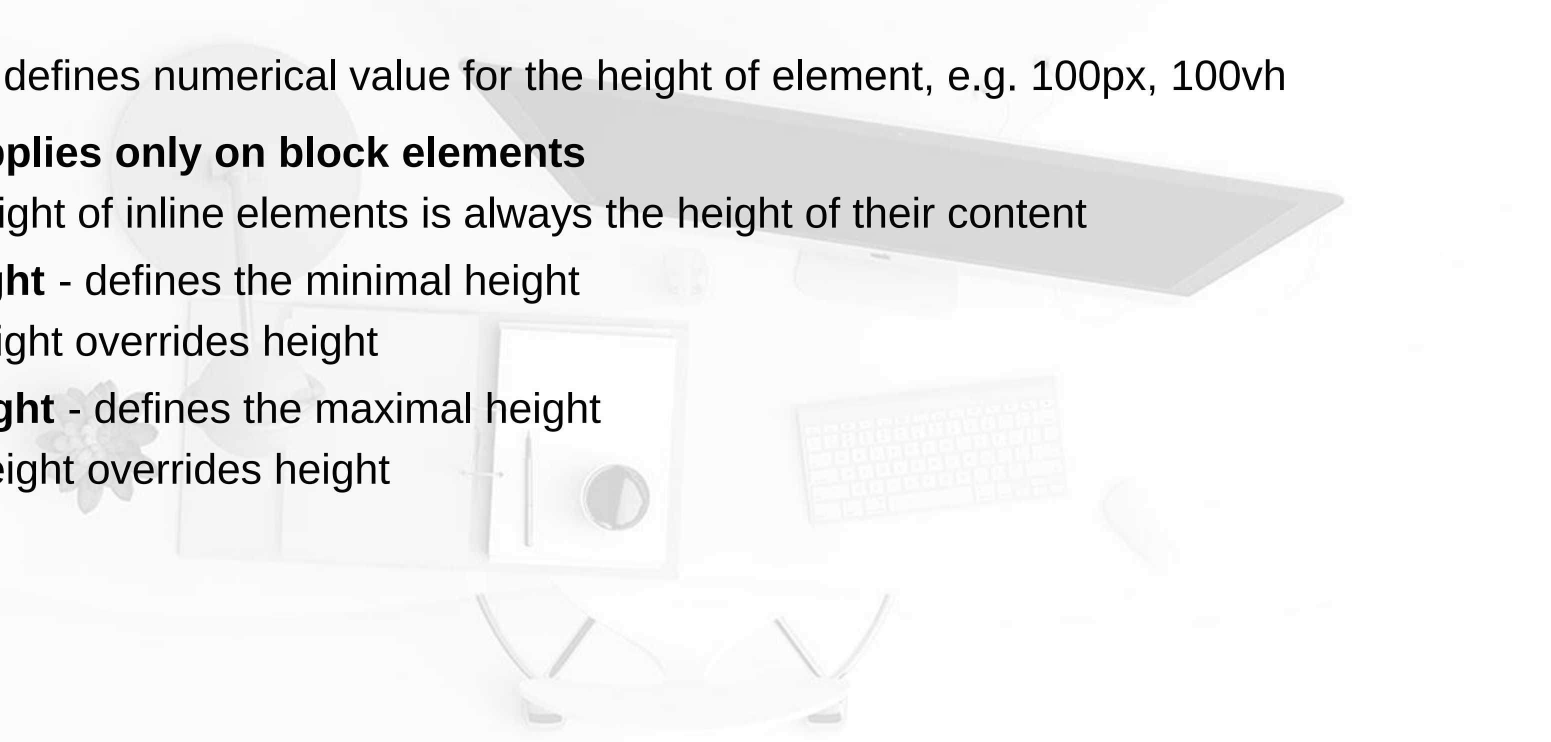
- **min-width** - defines the minimal width

  min-width overrides width if (width < min-width)

- **max-width** - defines the maximal width

  max-width overrides width if (width > max-width)

Height



- **height** – defines numerical value for the height of element, e.g. 100px, 100vh
- height **applies only on block elements**
  - The height of inline elements is always the height of their content
- **min-height** - defines the minimal height
  - min-height overrides height
- **max-height** - defines the maximal height
  - max-height overrides height

Width and Height Values

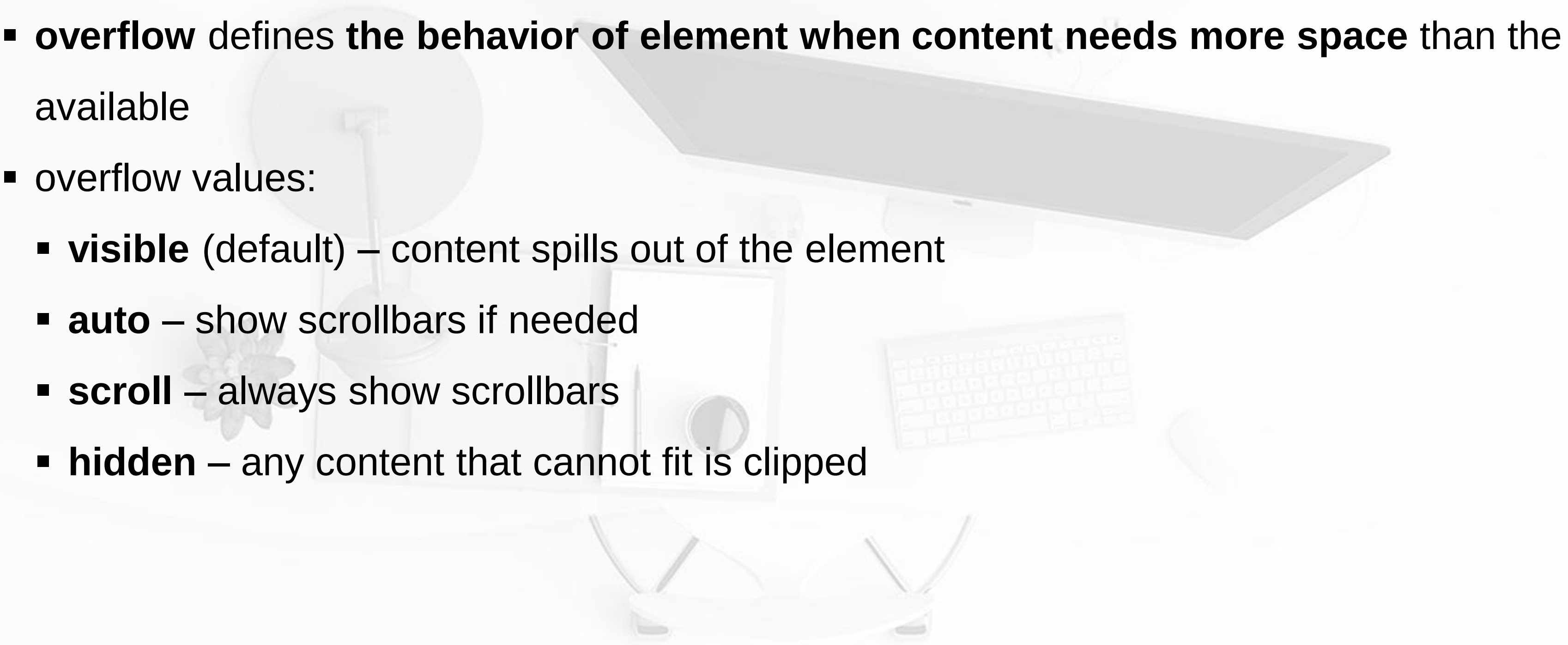
The values of the width and height properties are numerical:

- Pixels (px)
- Percentages

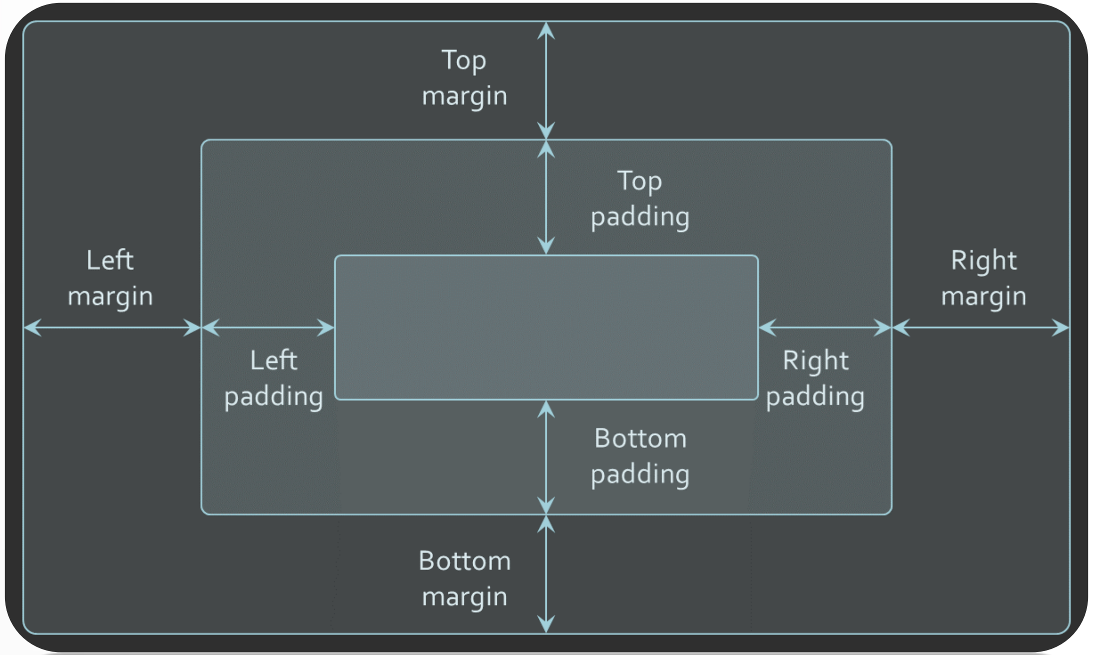
  A percent of the available width
- Centimeters (cm)

3. Overflow

Overflow

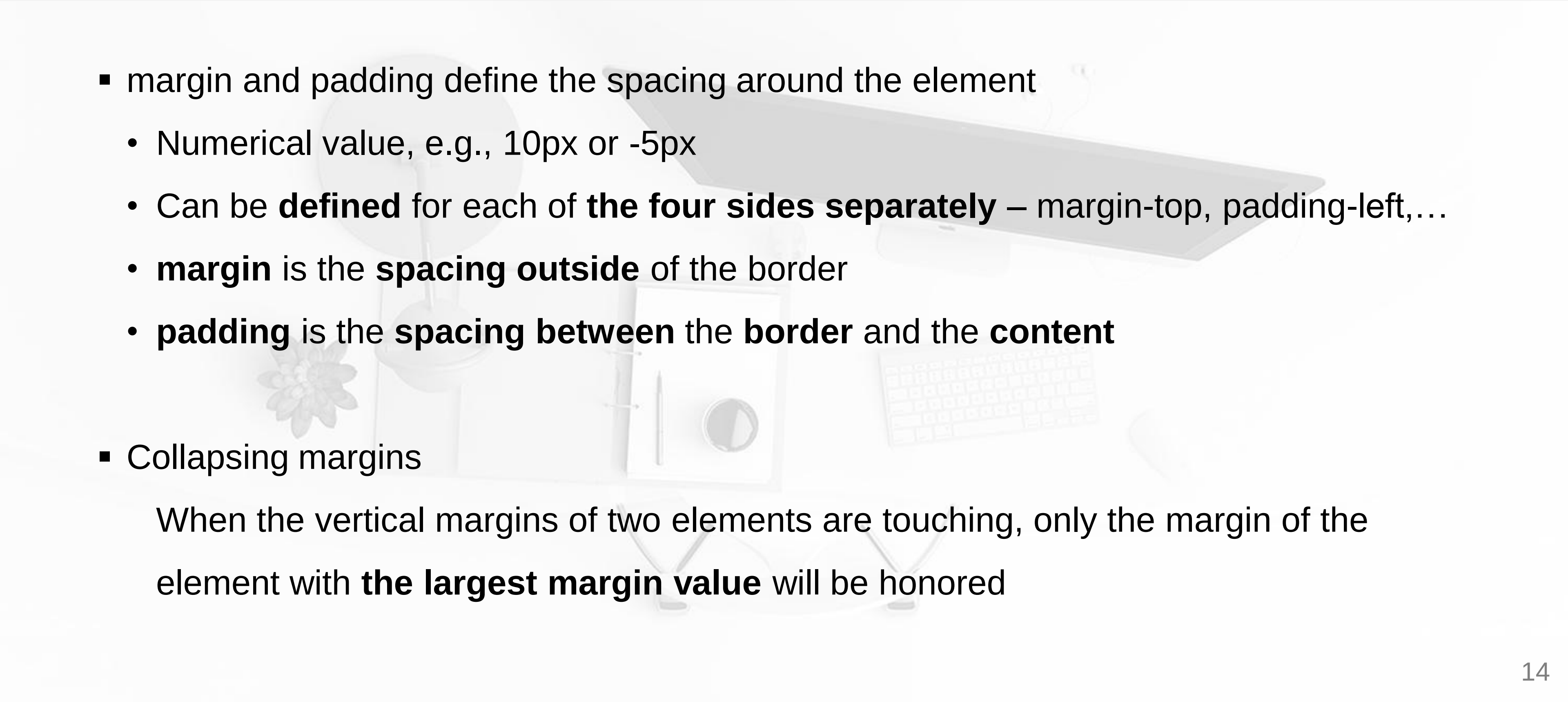


- **overflow** defines **the behavior of element when content needs more space** than the available
- overflow values:
  - **visible** (default) – content spills out of the element
  - **auto** – show scrollbars if needed
  - **scroll** – always show scrollbars
  - **hidden** – any content that cannot fit is clipped



4. Margin and Padding

Most Common Attributes

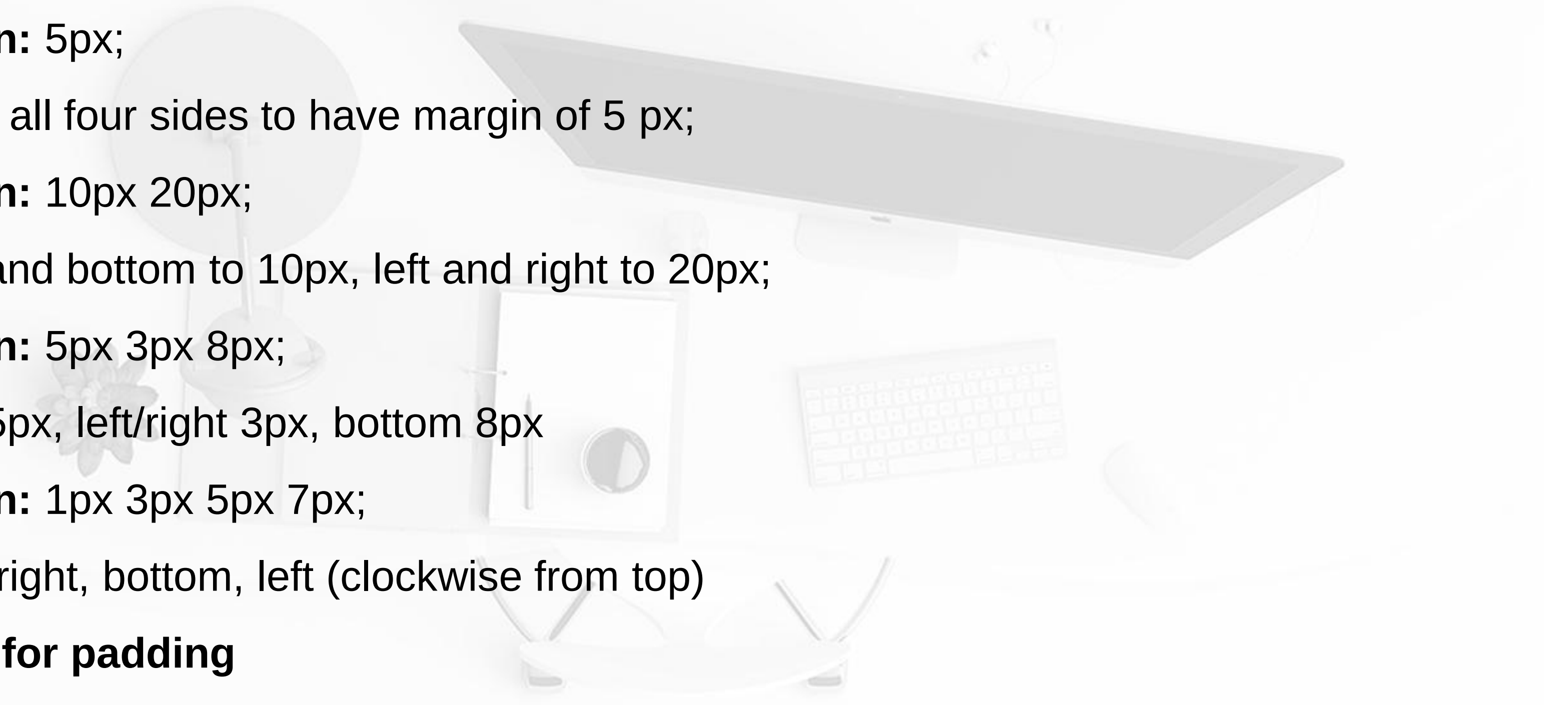


- margin and padding define the spacing around the element
  - Numerical value, e.g., 10px or -5px
  - Can be **defined** for each of **the four sides separately** – margin-top, padding-left,…
  - **margin** is the **spacing outside** of the border
  - **padding** is the **spacing between** the **border** and the **content**

- Collapsing margins

  When the vertical margins of two elements are touching, only the margin of the element with **the largest margin value** will be honored

Margin and Padding: Short Rules



- **margin:** 5px;

  Sets all four sides to have margin of 5 px;
- **margin:** 10px 20px;

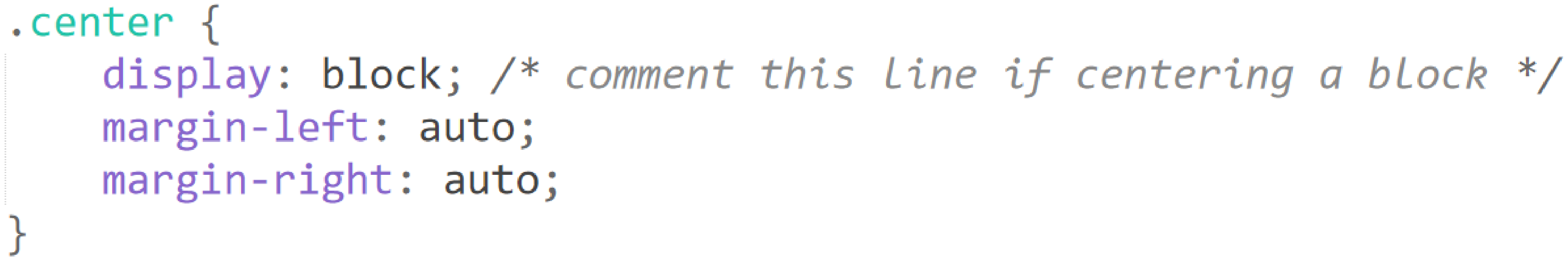
  top and bottom to 10px, left and right to 20px;
- **margin:** 5px 3px 8px;

  top 5px, left/right 3px, bottom 8px
- **margin:** 1px 3px 5px 7px;

  top, right, bottom, left (clockwise from top)
- **Same for padding**

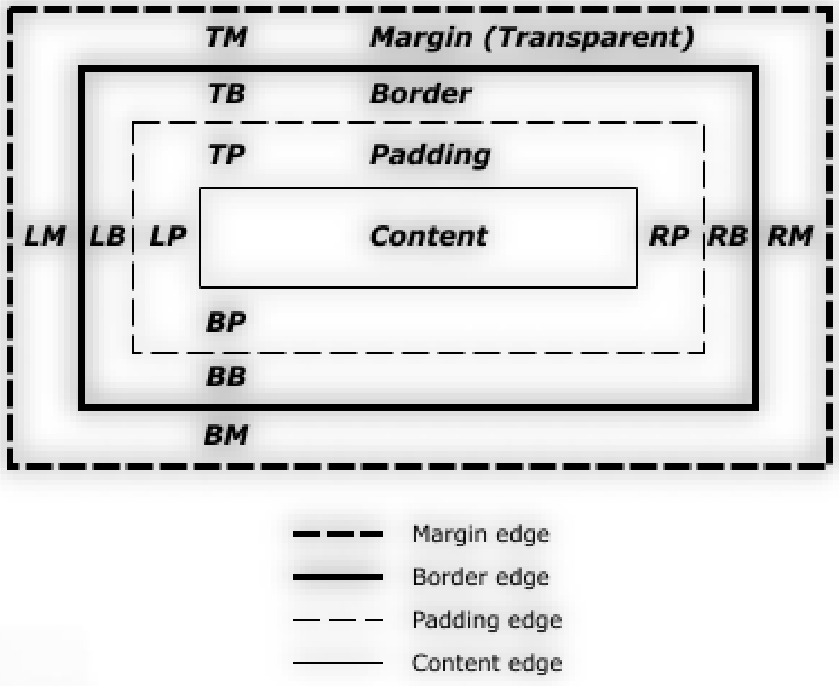
How To Center a block or an Image

Horizontal align



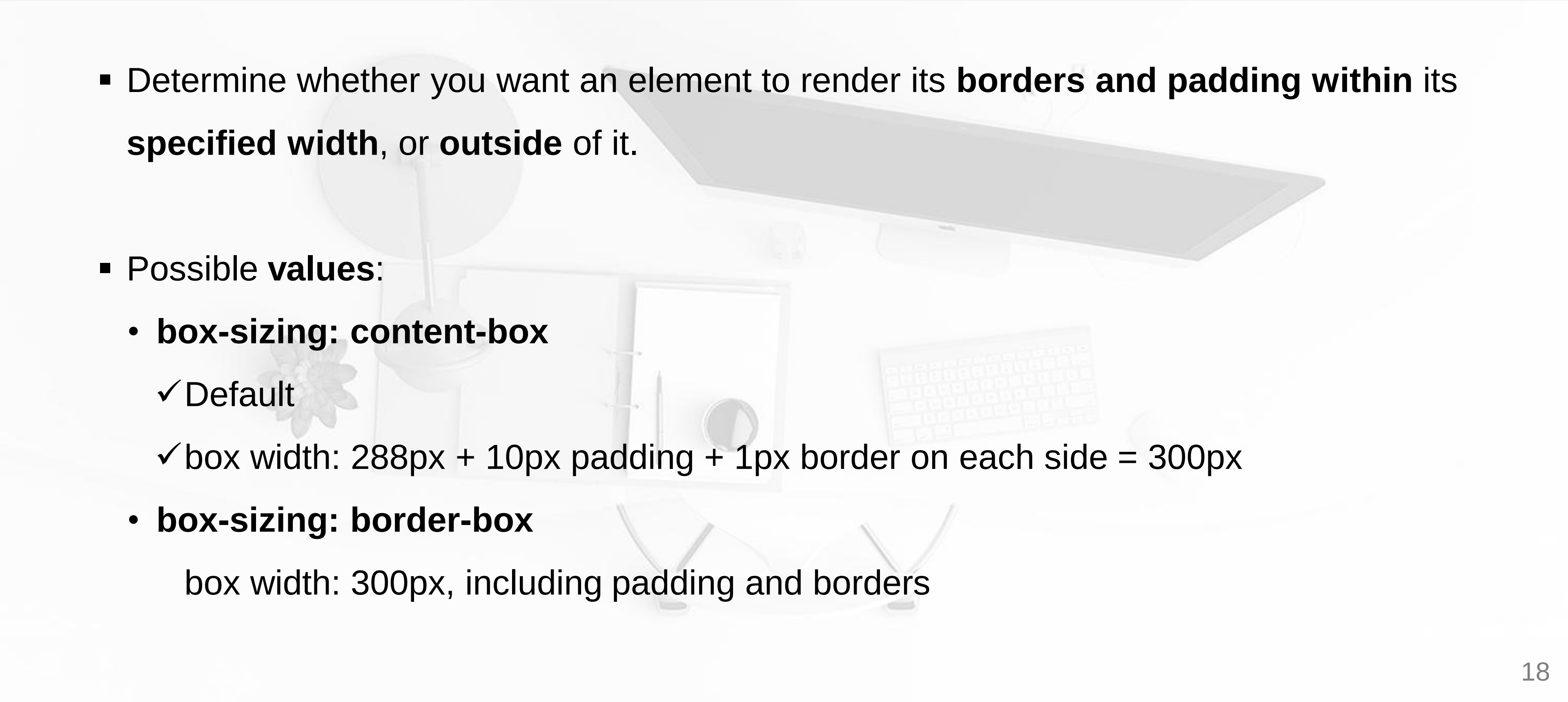
```css
.center {
    display: block; /* comment this line if centering a block */
    margin-left: auto;
    margin-right: auto;
}
```

5. Box Model


TM
Margin (Transparent)
TB
Border
TP
Padding
LM
LB
LP
Content
RP
RB
RM
BP
BB
BM
Margin edge
Border edge
Padding edge
Content edge

box-sizing

CSS 3



- Determine whether you want an element to render its **borders and padding within** its **specified width**, or **outside** of it.

- Possible **values**:
  - **box-sizing: content-box**

    ✓Default

    ✓box width: 288px + 10px padding + 1px border on each side = 300px
  - **box-sizing: border-box**

    box width: 300px, including padding and borders

box-sizing

Example

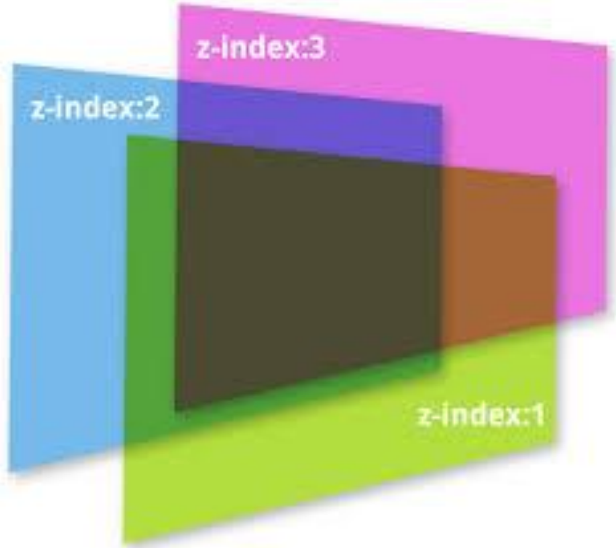
```
width: 300px;
border: 1px solid black;
padding: 5px;
box-sizing: border-box;

/* Firefox */
-moz-box-sizing: border-box;
/* WebKit */
-webkit-box-sizing: border-box;
```

Chuột Không Dây Logitech M187 - Hàng chính hãng

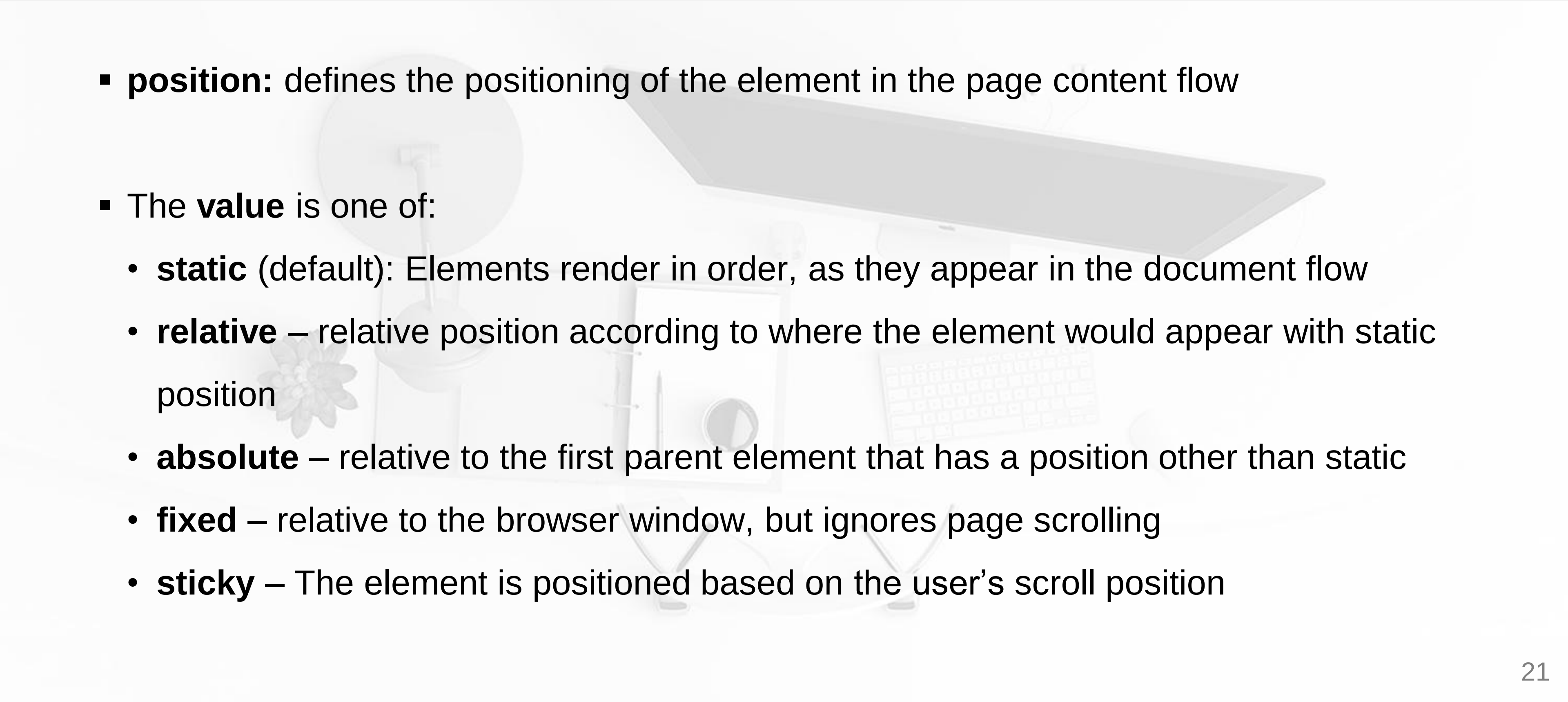
259.000 đ -19%

~~319.000 đ~~



6. Positioning

Positioning

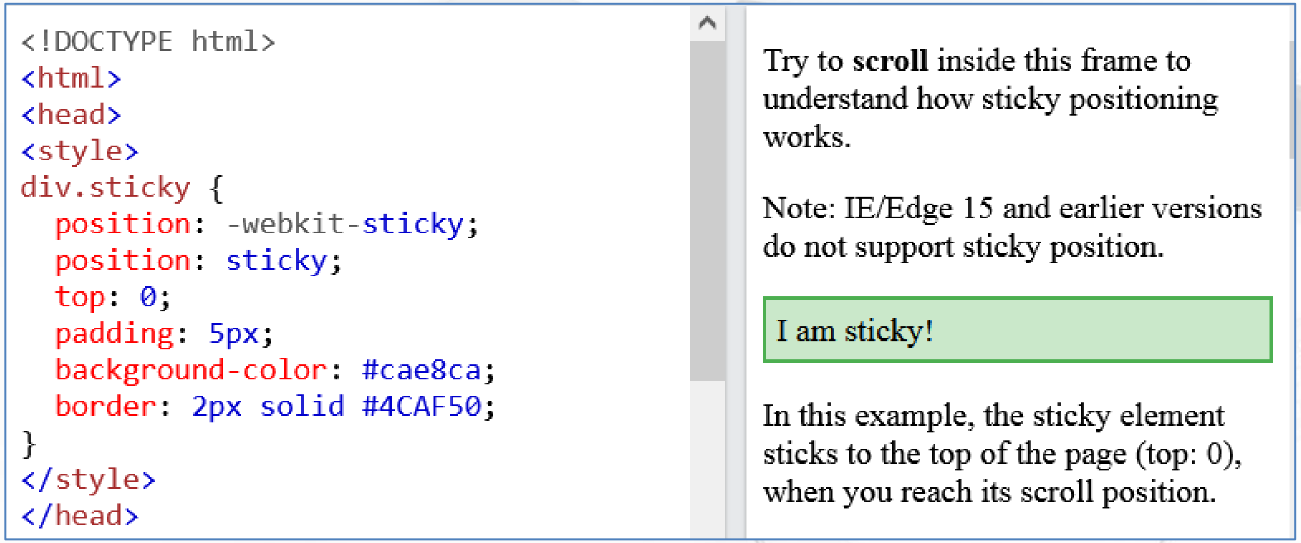


- **position:** defines the positioning of the element in the page content flow

- The **value** is one of:
  - **static** (default): Elements render in order, as they appear in the document flow
  - **relative –** relative position according to where the element would appear with static position
  - **absolute –** relative to the first parent element that has a position other than static
  - **fixed** – relative to the browser window, but ignores page scrolling
  - **sticky –** The element is positioned based on the user’s scroll position

sticky

Example



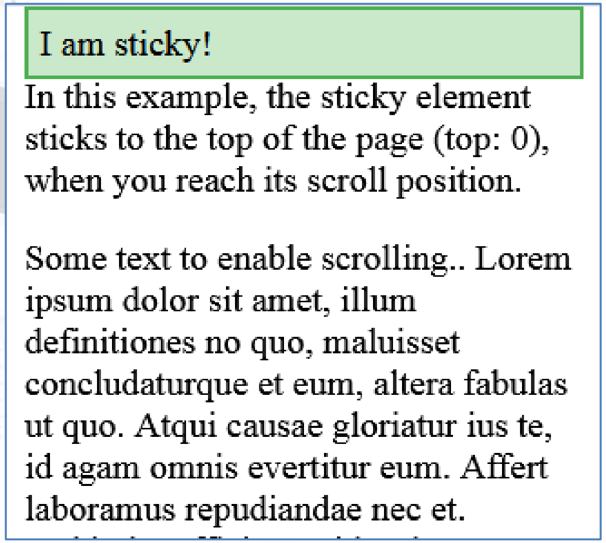
```
<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
</style>
</head>
```

Try to **scroll** inside this frame to understand how sticky positioning works.

Note: IE/Edge 15 and earlier versions do not support sticky position.

I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

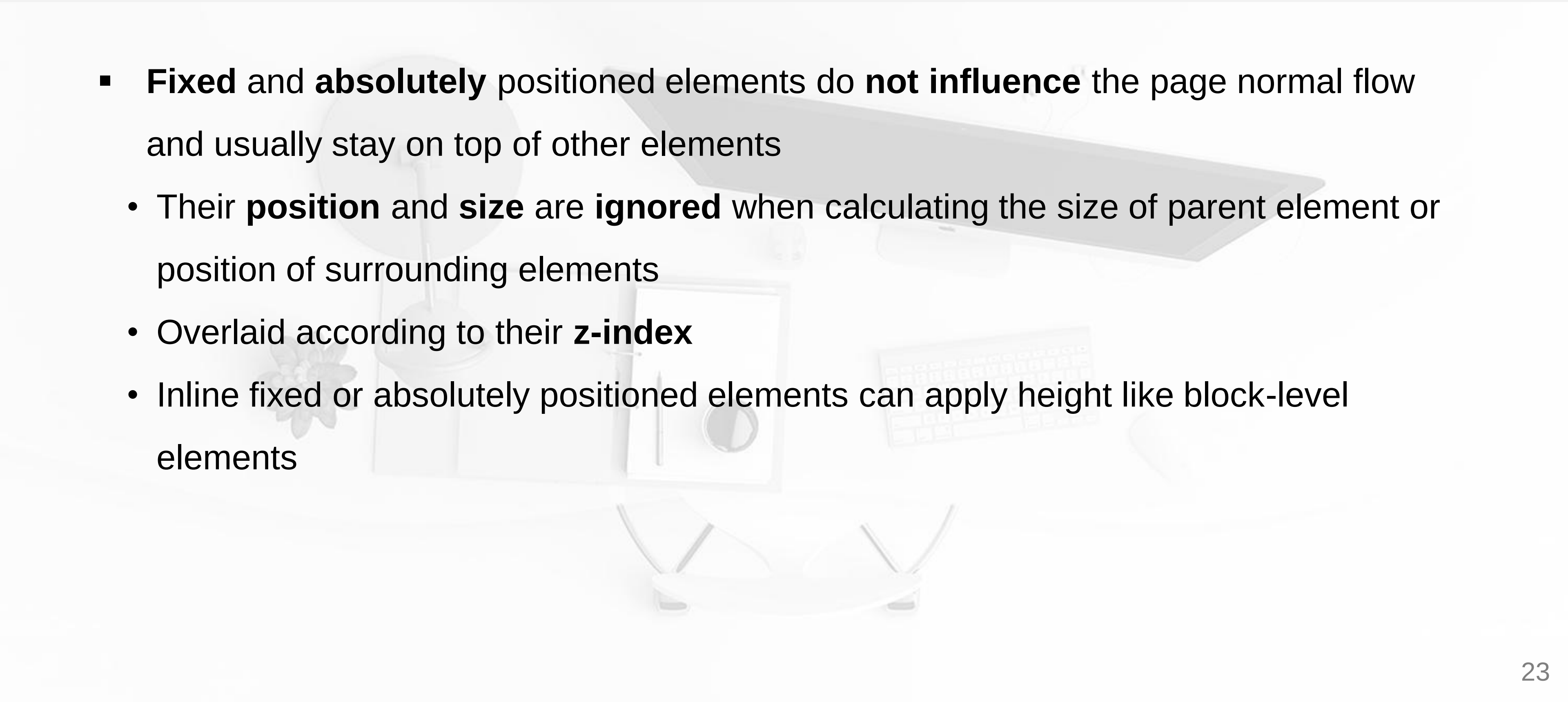


I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et.

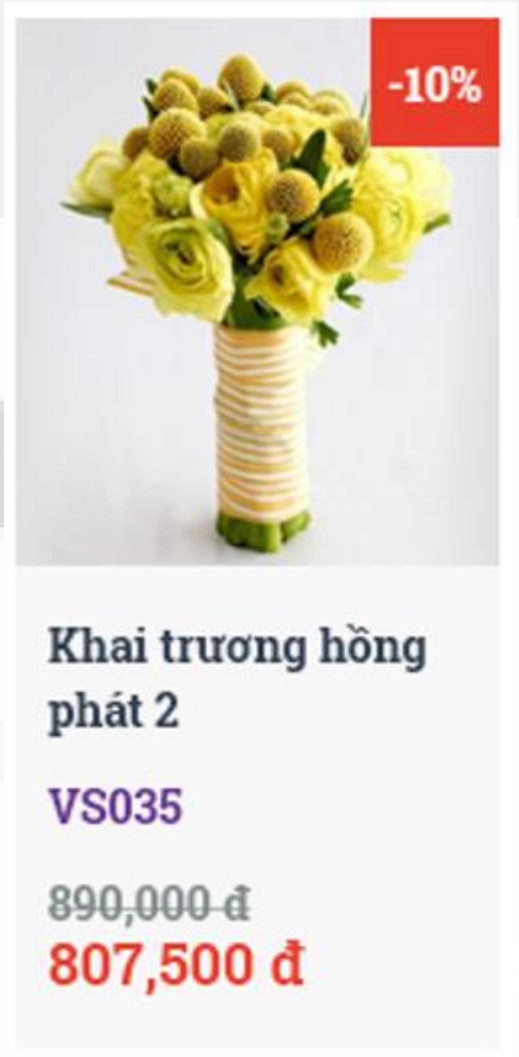
Positioning



- **Fixed** and **absolutely** positioned elements do **not influence** the page normal flow and usually stay on top of other elements
  - Their **position** and **size** are **ignored** when calculating the size of parent element or position of surrounding elements
  - Overlaid according to their **z-index**
  - Inline fixed or absolutely positioned elements can apply height like block-level elements

Demo

Create a discount section for the following design



-10%

Khai trương hồng phát 2

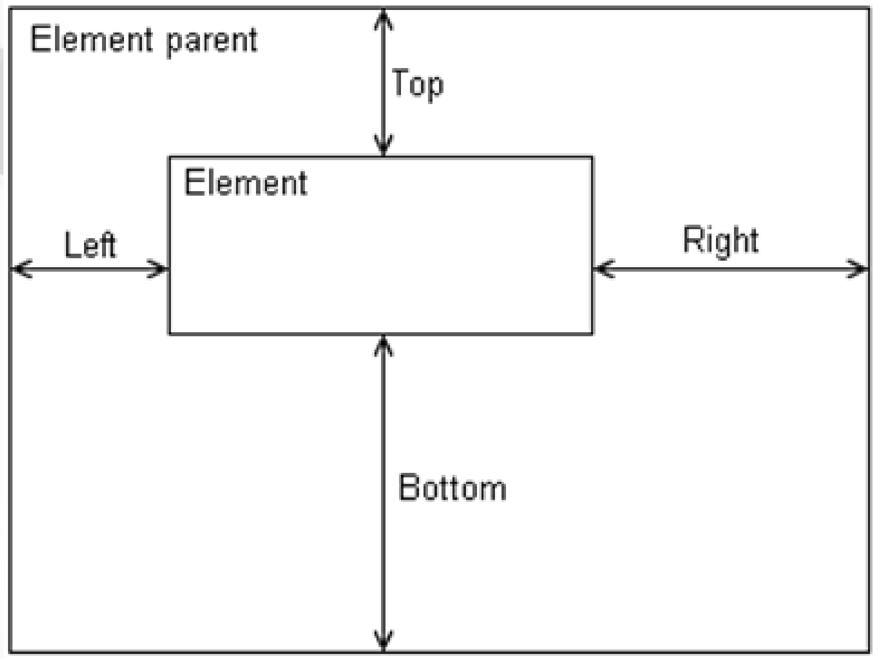
VS035

~~890,000 đ~~

807,500 đ

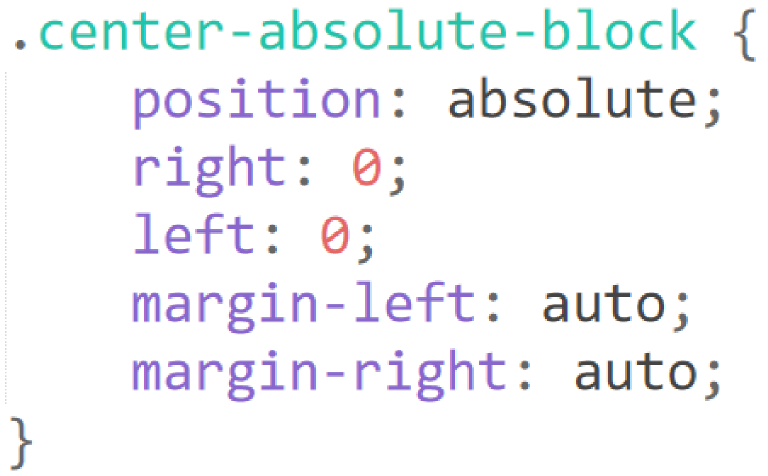
Positioning

top, left, bottom, right: specifies **offset of absolute/fixed/relative** positioned element as numerical values



How to center an absolute block

Horizontal align

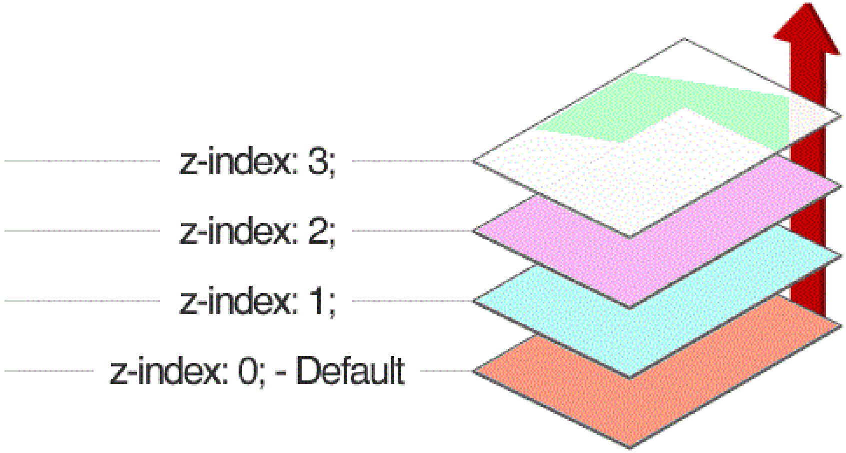


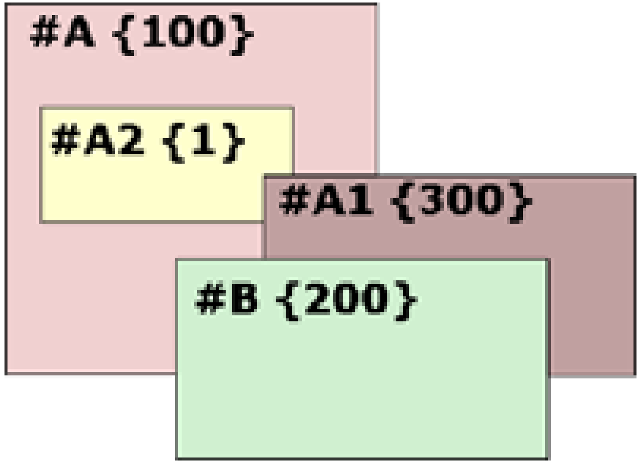
```css
.center-absolute-block {
    position: absolute;
    right: 0;
    left: 0;
    margin-left: auto;
    margin-right: auto;
}
```

Positioning

z-index: specifies the stack level of positioned elements

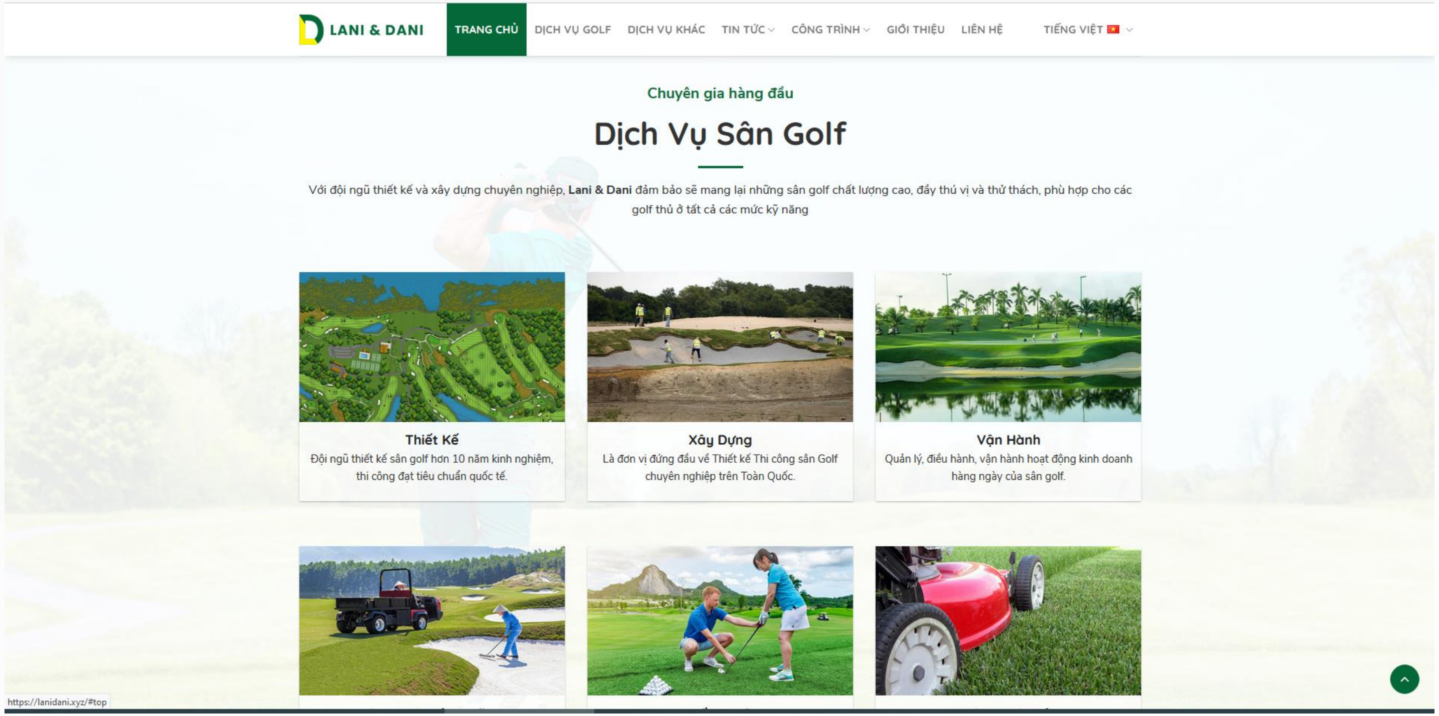
Understanding stacking context





Exercise

Create a “Go to top” button



LANI & DANI
TRANG CHỦ DỊCH VỤ GOLF DỊCH VỤ KHÁC TIN TỨC CÔNG TRÌNH GIỚI THIỆU LIÊN HỆ TIẾNG VIỆT

Chuyên gia hàng đầu

Dịch Vụ Sân Golf

Với đội ngũ thiết kế và xây dựng chuyên nghiệp, **Lani & Dani** đảm bảo sẽ mang lại những sân golf chất lượng cao, đầy thú vị và thử thách, phù hợp cho các golf thủ ở tất cả các mức kỹ năng

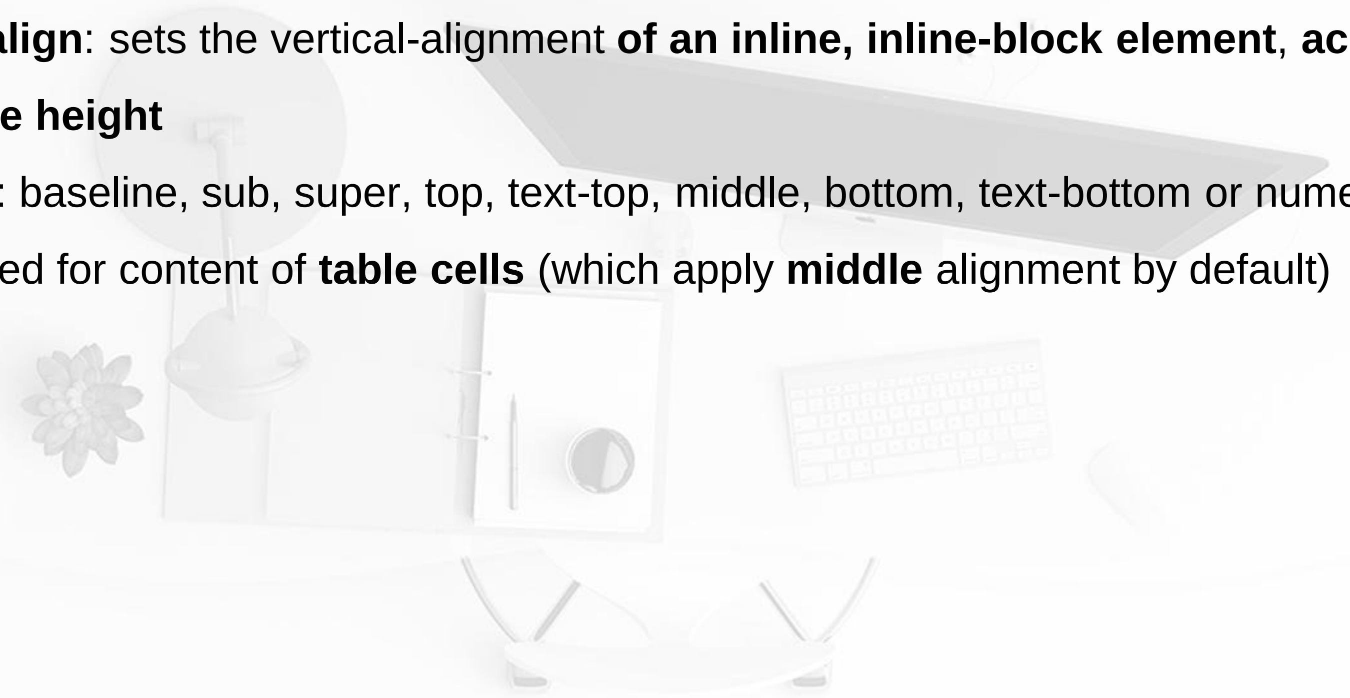
Thiết Kế
Đội ngũ thiết kế sân golf hơn 10 năm kinh nghiệm, thi công đạt tiêu chuẩn quốc tế.

Xây Dựng
Là đơn vị đứng đầu về Thiết kế Thi công sân Golf chuyên nghiệp trên Toàn Quốc.

Vận Hành
Quản lý, điều hành, vận hành hoạt động kinh doanh hàng ngày của sân golf.

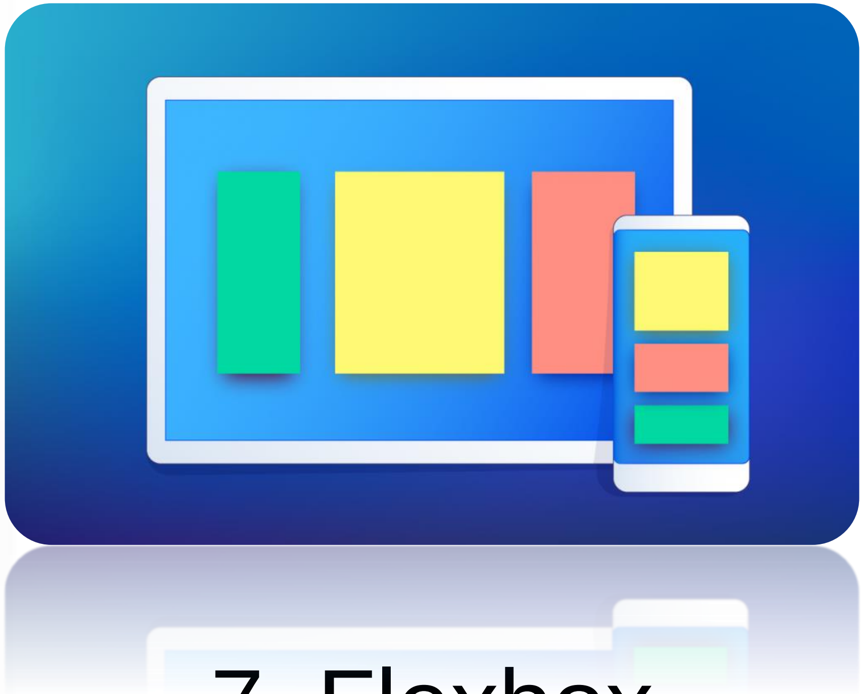
https://lanidani.xyz/#top

Inline element positioning



vertical-align: sets the vertical-alignment **of an inline, inline-block element**, **according to the line height**

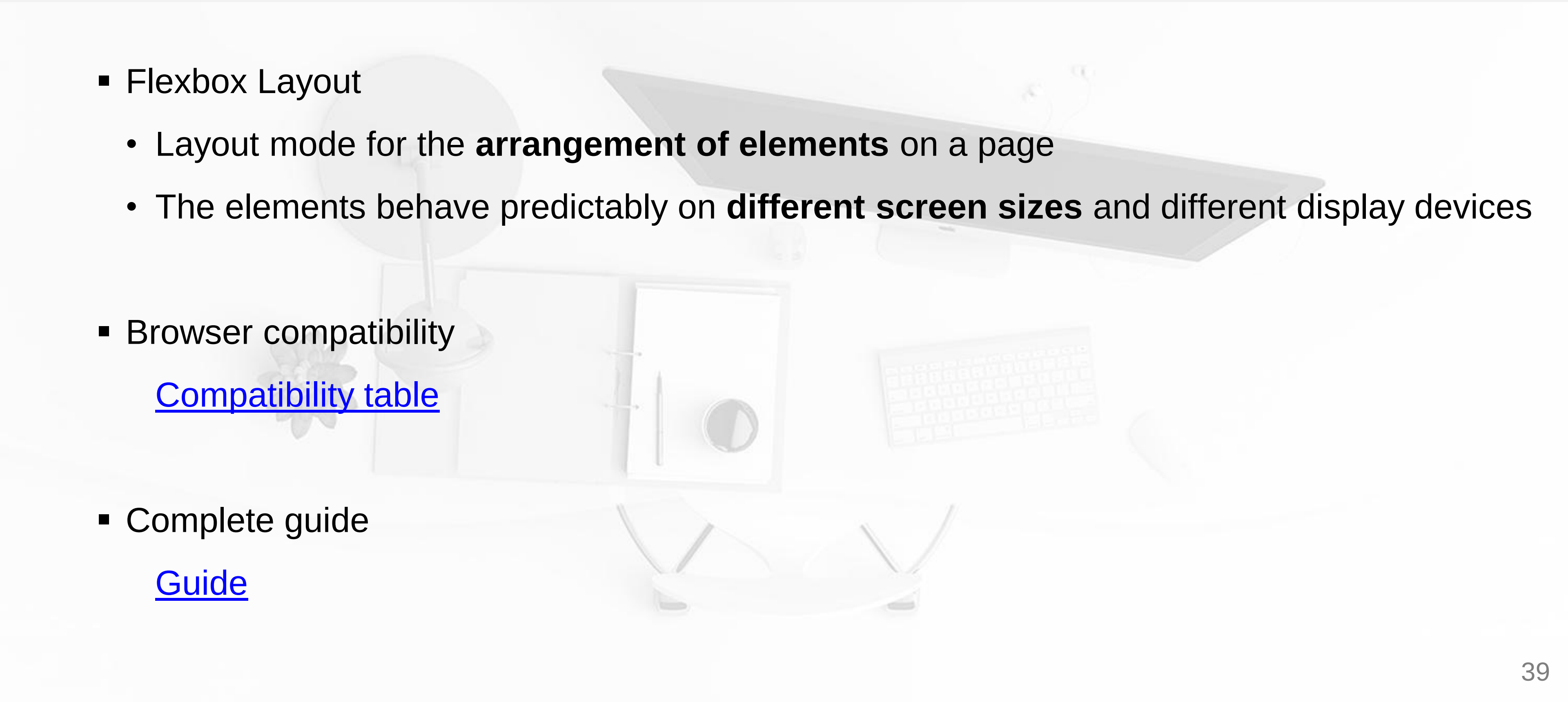
- **Values**: baseline, sub, super, top, text-top, middle, bottom, text-bottom or numeric
- Also used for content of **table cells** (which apply **middle** alignment by default)



7. Flexbox

The Next Generation of CSS Layout

Flexbox



- Flexbox Layout
  - Layout mode for the **arrangement of elements** on a page
  - The elements behave predictably on **different screen sizes** and different display devices

- Browser compatibility

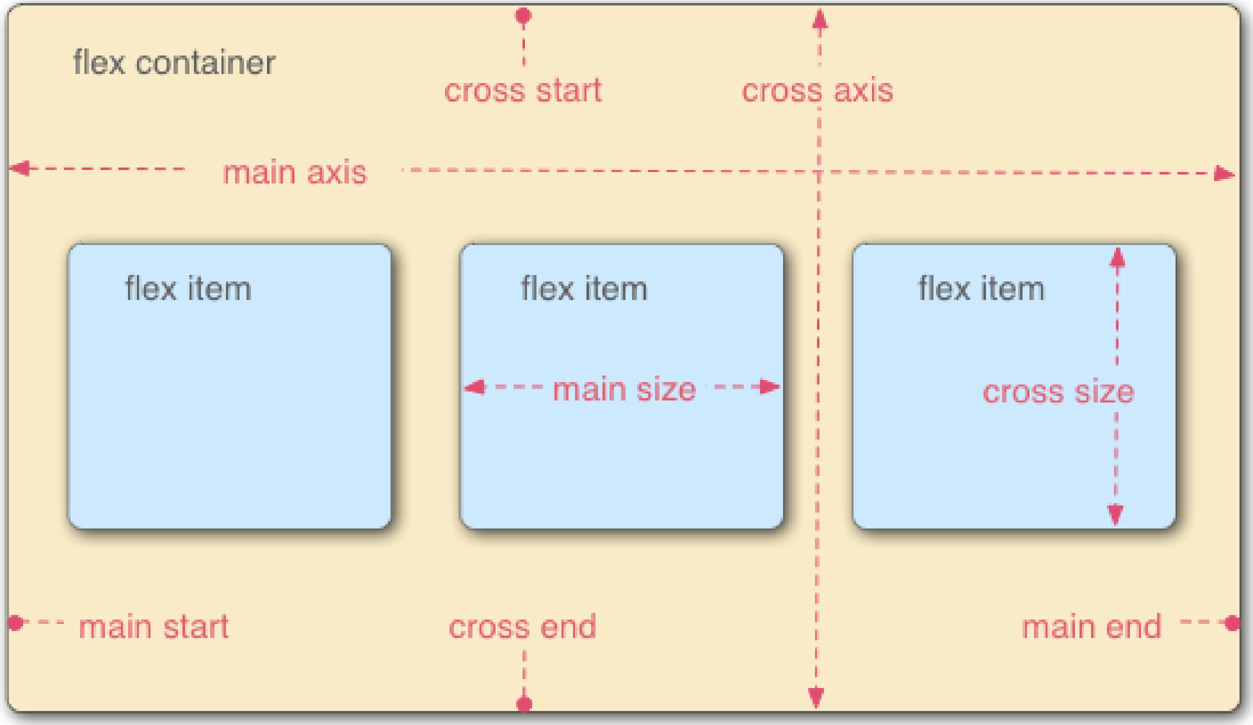
  Compatibility table

- Complete guide

  Guide

Flexbox vocabulary

- Flex container
- Flex item
- Axes
  - Main axis
  - Cross axis
- Directions
- Lines
- Dimensions



Parent properties

- **display** - enables flex for all children

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

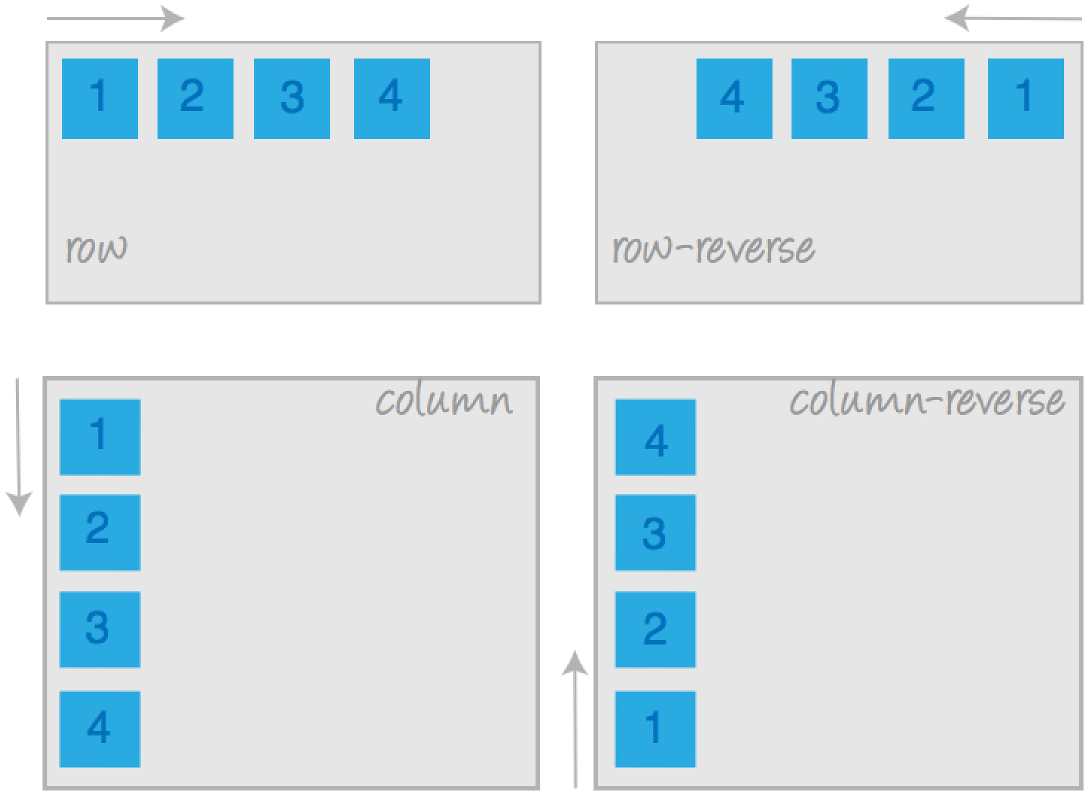
- **flex-direction** - establishes the main-axis

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

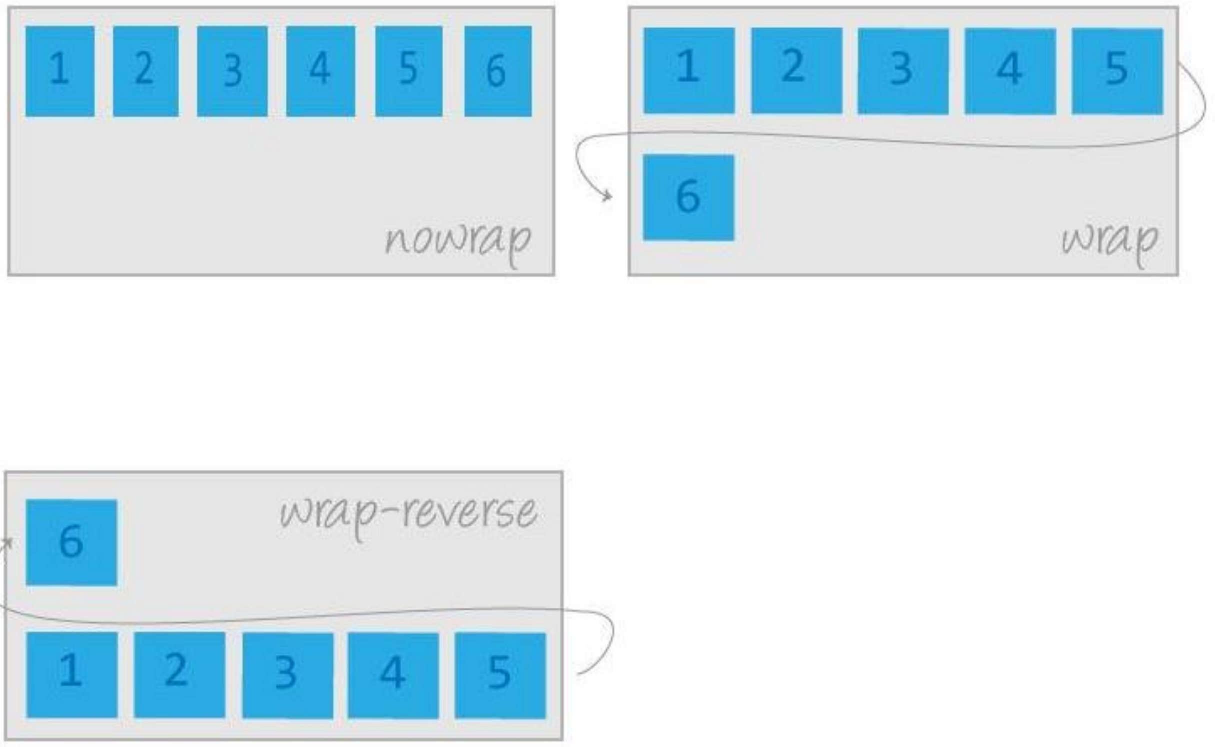
- **flex-wrap** - flex items will all try to fit onto one line by default

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

Flex-direction



Flex-wrap



Parent properties

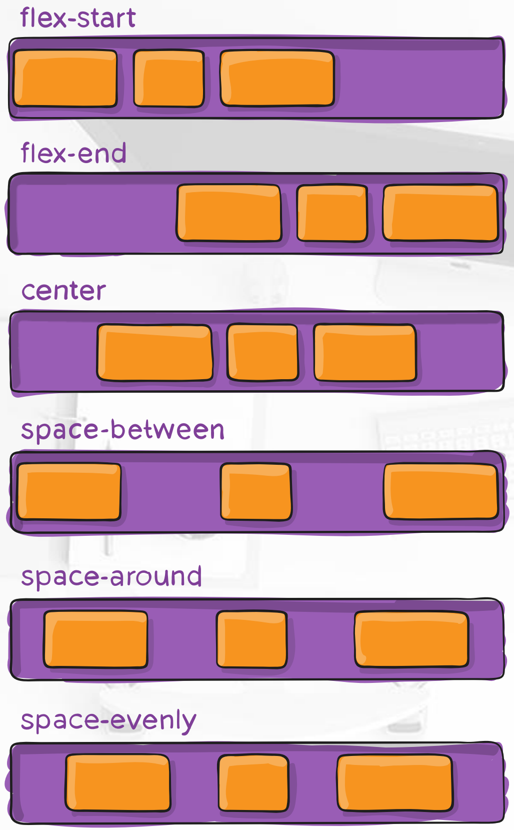
- **flex-flow** - shorthand for flex-direction and flex-wrap

```
.container {
    flex-flow: <‘flex-direction’> <‘flex-wrap’>
}
```

- **justify-content** - align the items on the main axis

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

Justify-content



Parent properties

- **align-items** - align the items on the cross axis

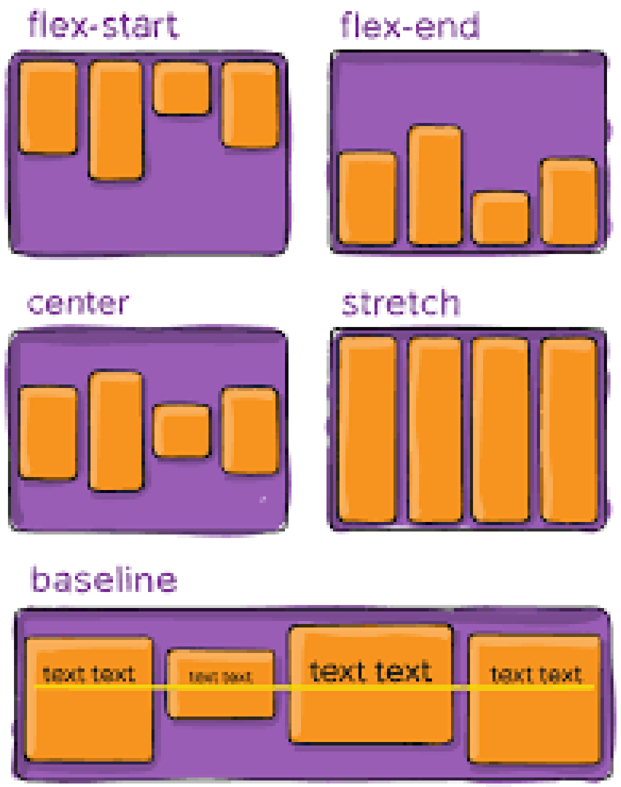
```
.container {
    align-items: flex-start | flex-end |
        center | baseline | stretch;
}
```

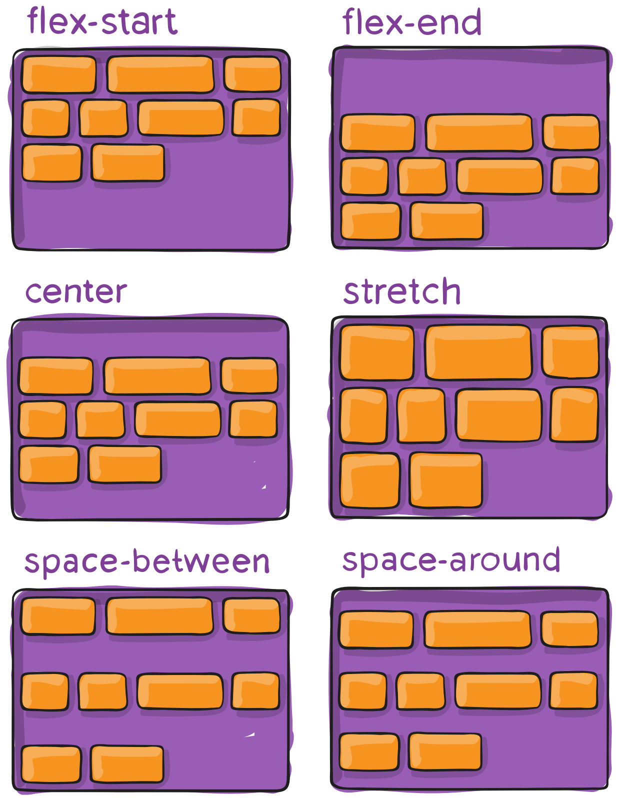
- **align-content** - flex container’s lines within when there is extra space in the cross-axis

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

Align-items

Align-content





Children properties

- **order** - controls the order in which the children appear in the flex container

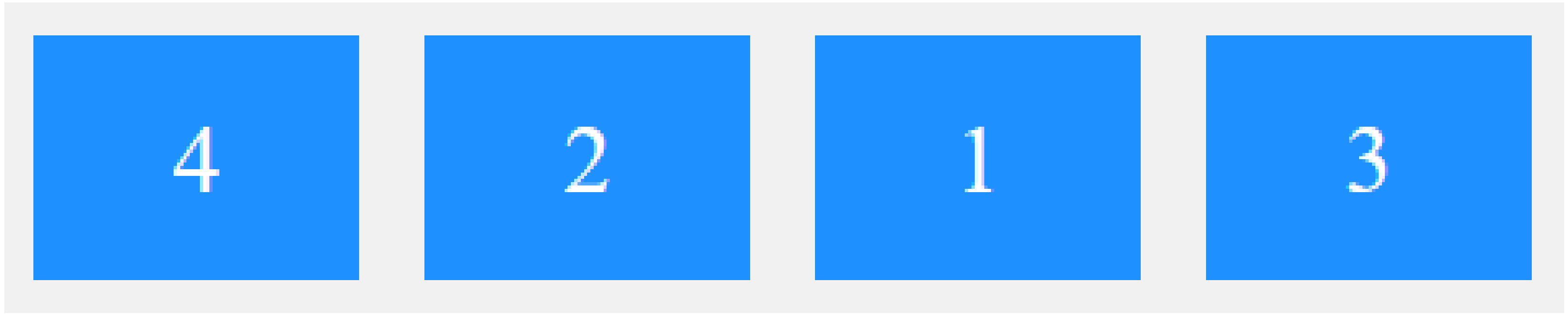
```
.item {
    order: <integer>;
}
```

- **flex-grow** - defines the ability for a flex item to grow if necessary

```
.item {
    flex-grow: <number>; /* default 0 */
}
```

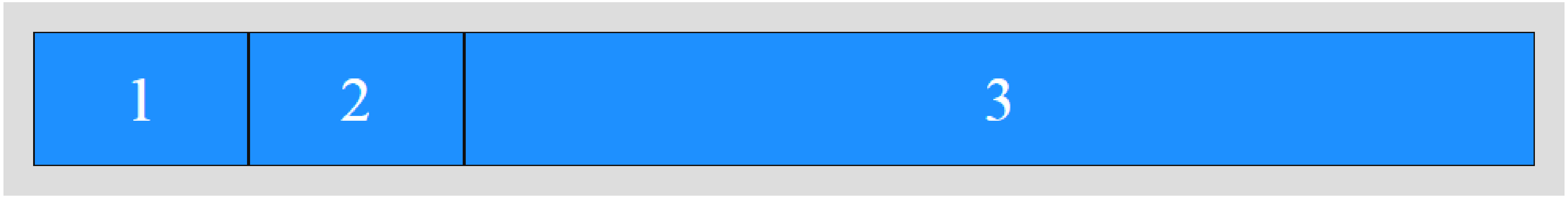
Order

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```



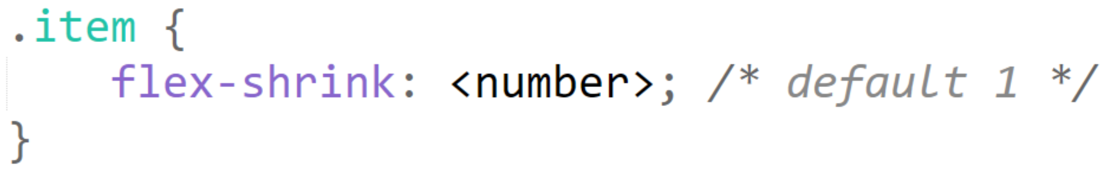
flex-grow

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```



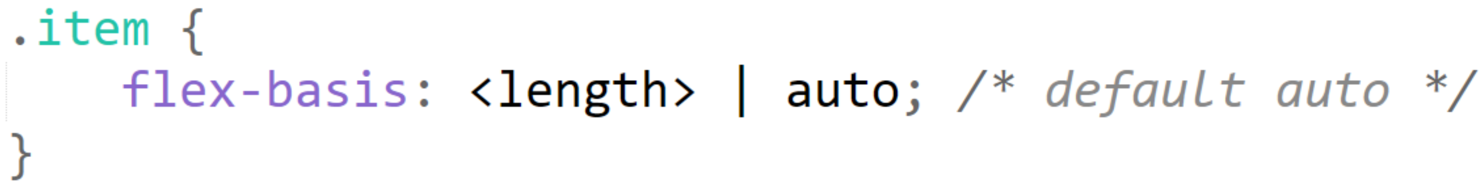
Children properties

- **flex-shrink** - defines the ability for a flex item to shrink if necessary



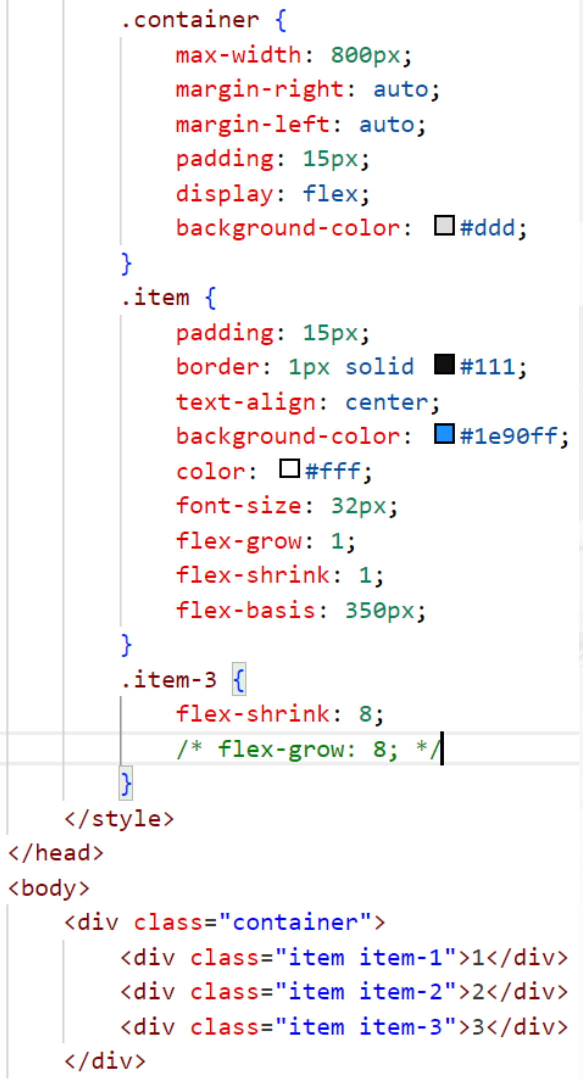
```
.item {
    flex-shrink: <number>; /* default 1 */
}
```

- **flex-basis** - defines the default size of an element before the remaining space is distributed

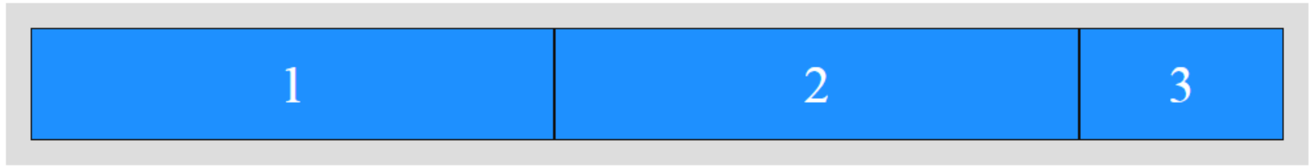


```
.item {
    flex-basis: <length> | auto; /* default auto */
}
```

Flex-shrink



```
    .container {
        max-width: 800px;
        margin-right: auto;
        margin-left: auto;
        padding: 15px;
        display: flex;
        background-color: #ddd;
    }
    .item {
        padding: 15px;
        border: 1px solid #111;
        text-align: center;
        background-color: #1e90ff;
        color: #fff;
        font-size: 32px;
        flex-grow: 1;
        flex-shrink: 1;
        flex-basis: 350px;
    }
    .item-3 {
        flex-shrink: 8;
        /* flex-grow: 8; */
    }
</style>
</head>
<body>
<div class="container">
    <div class="item item-1">1</div>
    <div class="item item-2">2</div>
    <div class="item item-3">3</div>
</div>
```



Children properties

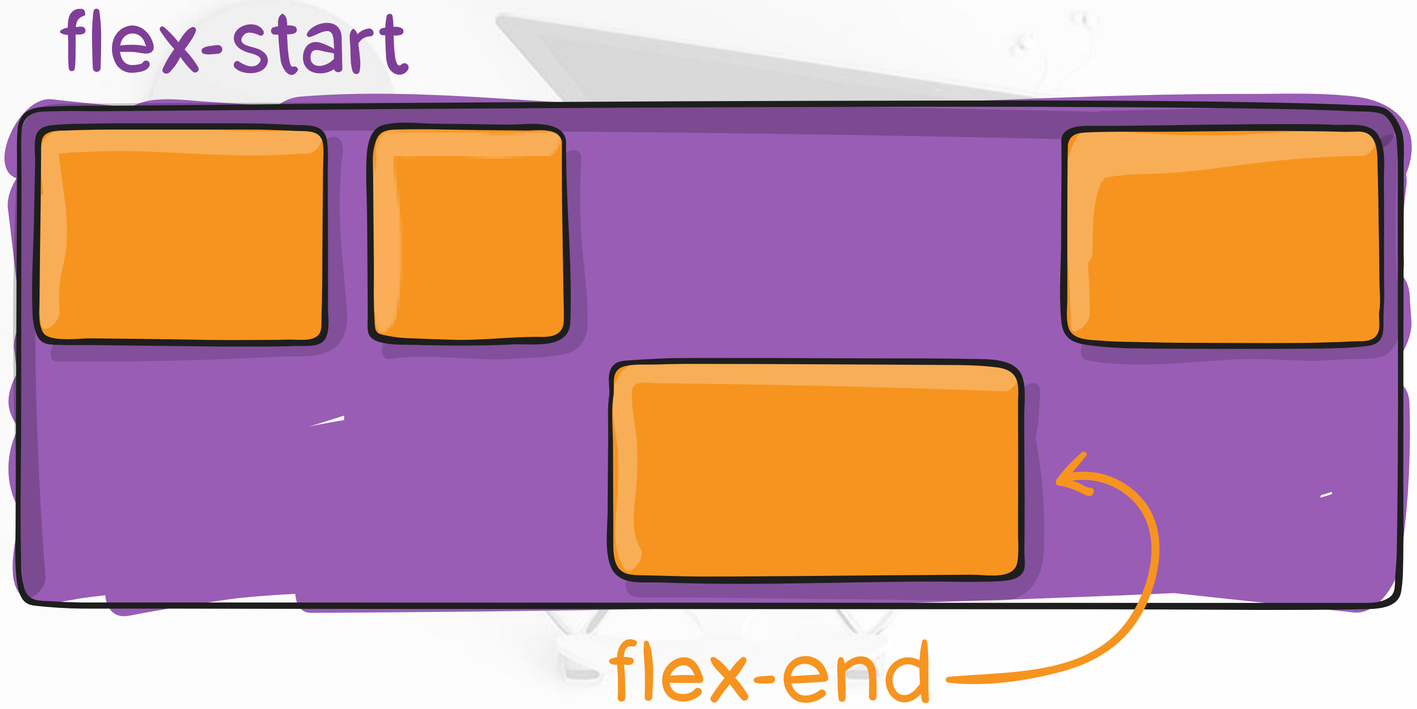
- **flex** - shorthand for **flex-grow**, **flex-shrink** and **flex-basis** combined (**recommended**)

```
.item {
    flex: none |
        [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

- **align-self** - allows the default alignment (or the one specified by align-items) to be overridden for individual flex items

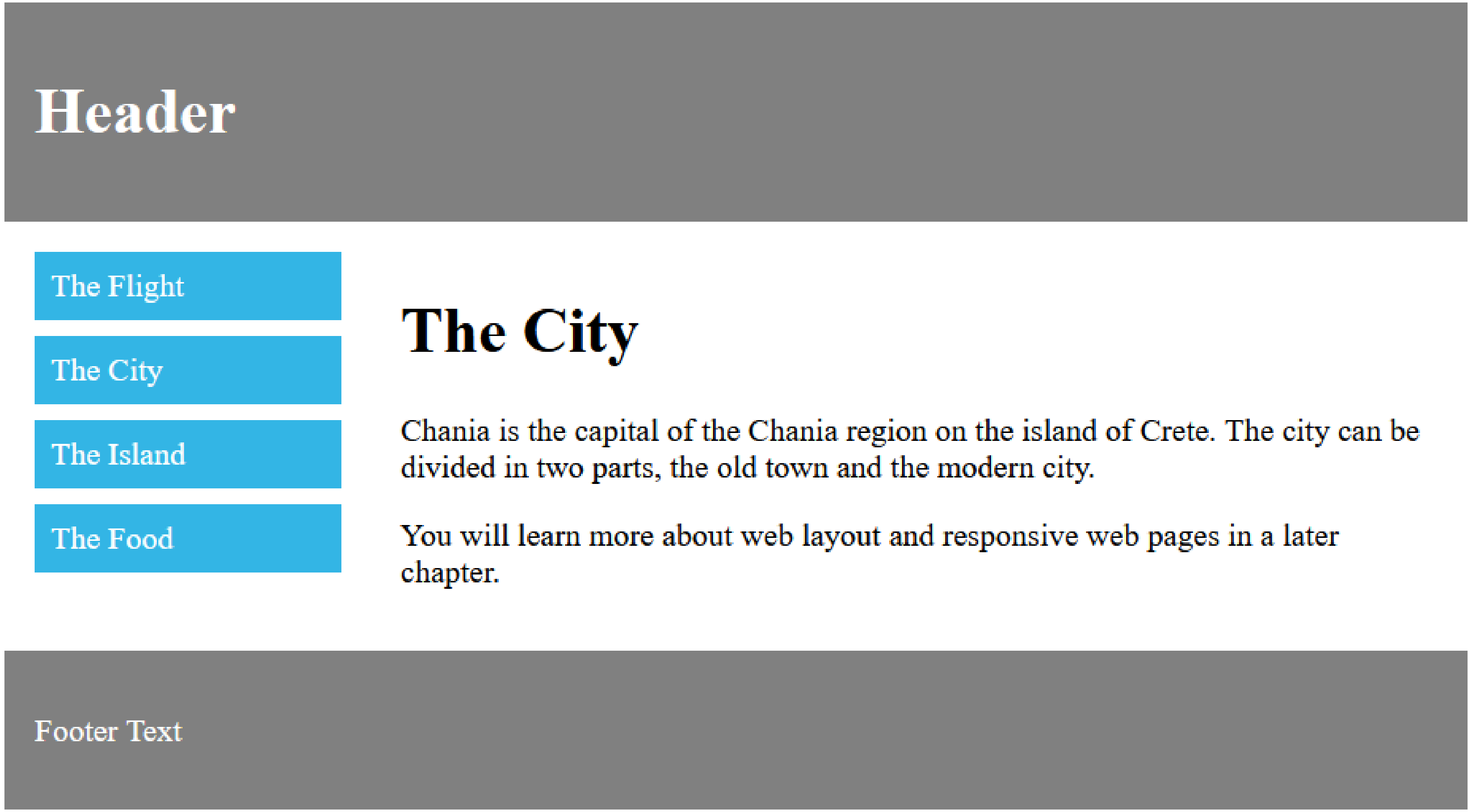
```
.item {
    align-self: auto | flex-start | flex-end |
        center | baseline | stretch;
}
```

Align-self



CSS – Layout

Exercise



Header

The Flight

The City

The Island

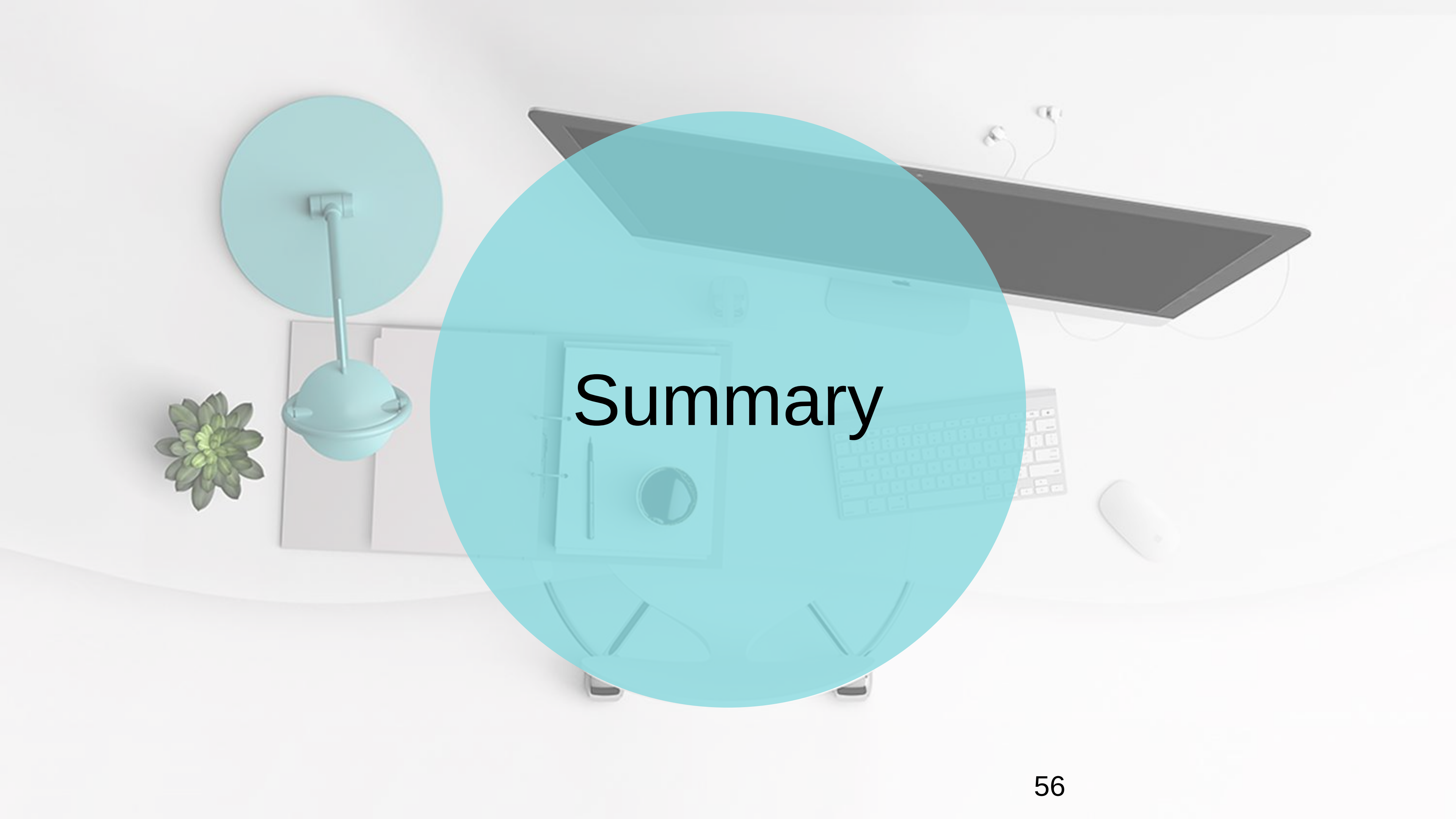
The Food

The City

Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.

You will learn more about web layout and responsive web pages in a later chapter.

Footer Text



Summary

Exercise

Complete the Lani & Dani webpage (only Desktop UI).

Q&A