



# Positioning Blocks on the Page and **Flexbox**

Lesson 7



# Lesson Plan

1

Document Flow

2

Box Model

3

Positioning in the Flow

4

Flexbox

# document flow

- is an abstract model of how elements on a webpage are arranged and interact with each other. It determines the order in which elements will be displayed on the page.
- The standard flow runs from left to right, top to bottom. The flow is maintained in any element.
- The flow consists of two types of elements:
  - **inline**
  - **block**

# Content

**block-level boxes** – Occupy the entire available horizontal width and start on a new line

`display: block, list-item, table, flex, grid`

<code>&lt;h1&gt;-&lt;h6&gt;</code>	<code>&lt;nav&gt;</code>	<code>&lt;main&gt;</code>	<code>&lt;header&gt;</code>
<code>&lt;div&gt;</code>	<code>&lt;section&gt;</code>	<code>&lt;form&gt;</code>	<code>&lt;footer&gt;</code>
<code>&lt;p&gt;</code>	<code>&lt;article&gt;</code>	<code>&lt;ol&gt; / &lt;ul&gt;</code>	<code>&lt;aside&gt;</code>

**inline-level boxes** – Occupy only the necessary horizontal space and are placed on the same line. They don't start on a new line and take up only as much space as they need.

`display: inline, inline-block, inline-table, inline-flex, inline-grid`

<code>&lt;a&gt;</code>	<code>&lt;label&gt;</code>	<code>&lt;select&gt;</code>	<code>&lt;button&gt;</code>
<code>&lt;span&gt;</code>	<code>&lt;textarea&gt;</code>	<code>&lt;input&gt;</code>	<code>&lt;img&gt;</code>

# document flow

Normal flow



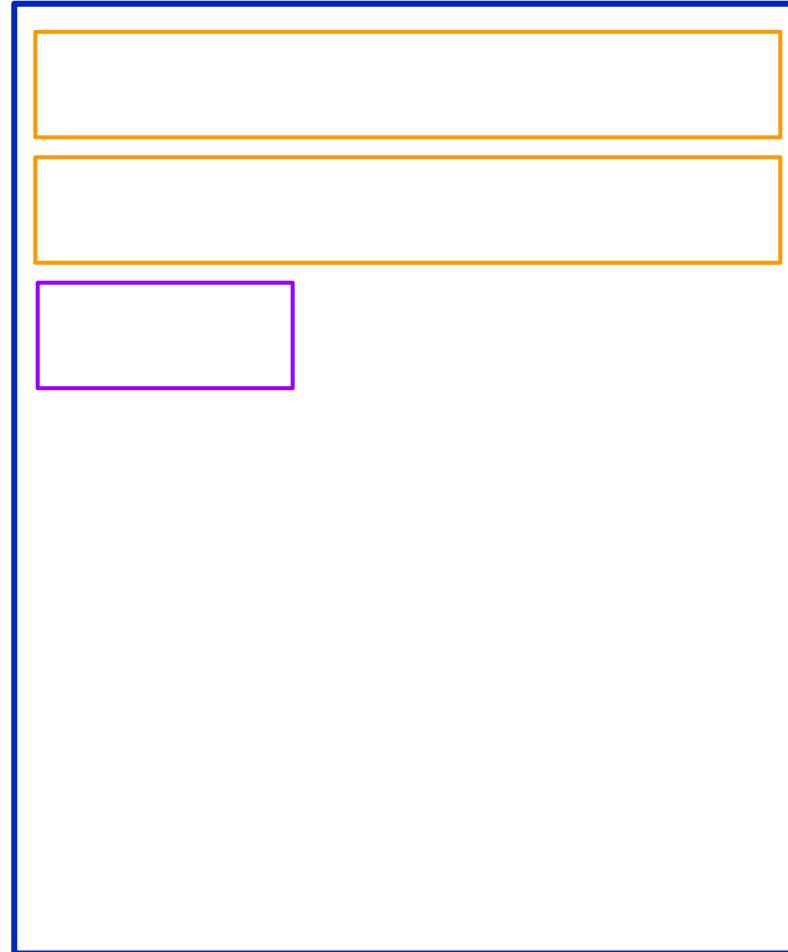
# document flow

Normal flow



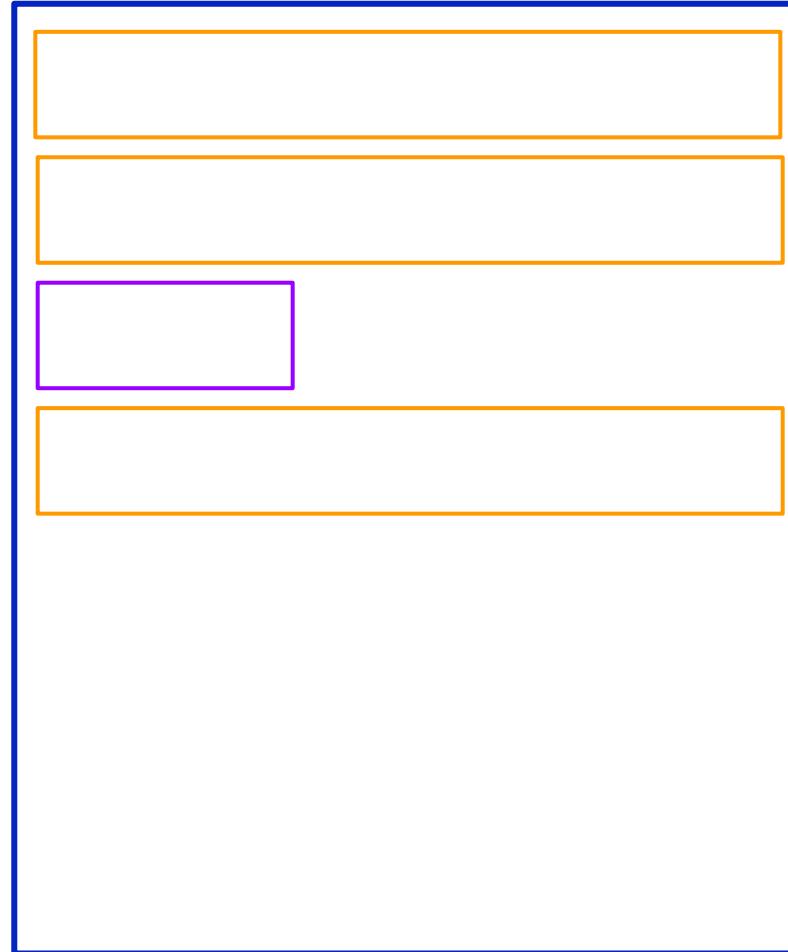
# document flow

Normal flow



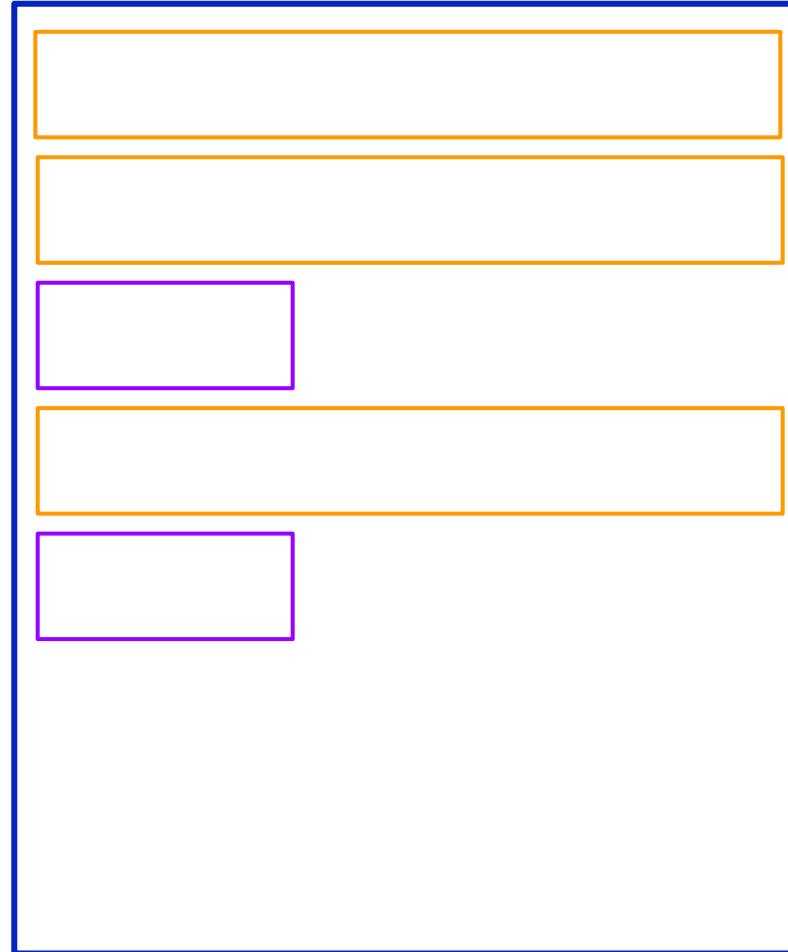
# document flow

Normal flow



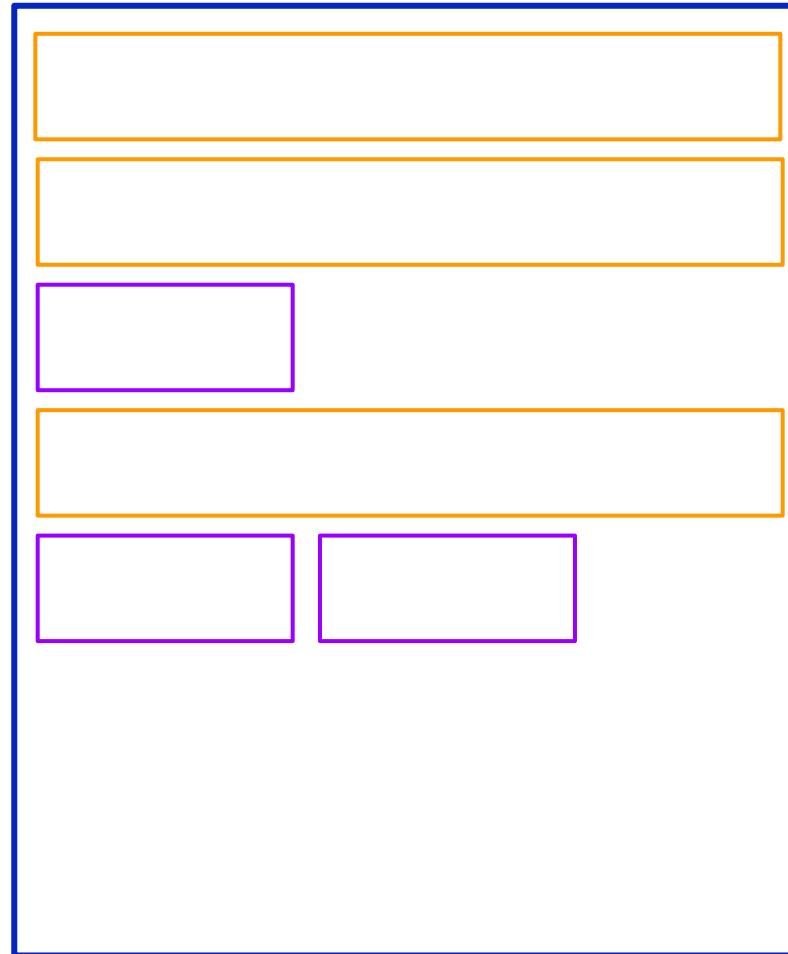
# document flow

Normal flow



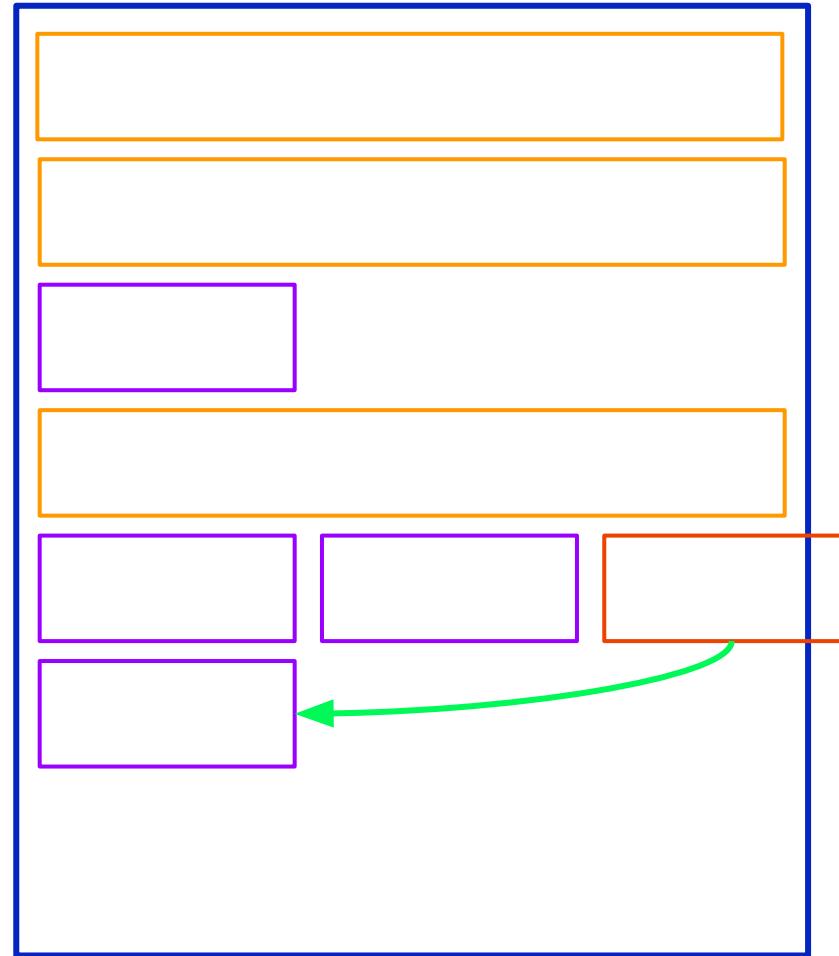
# document flow

Normal flow



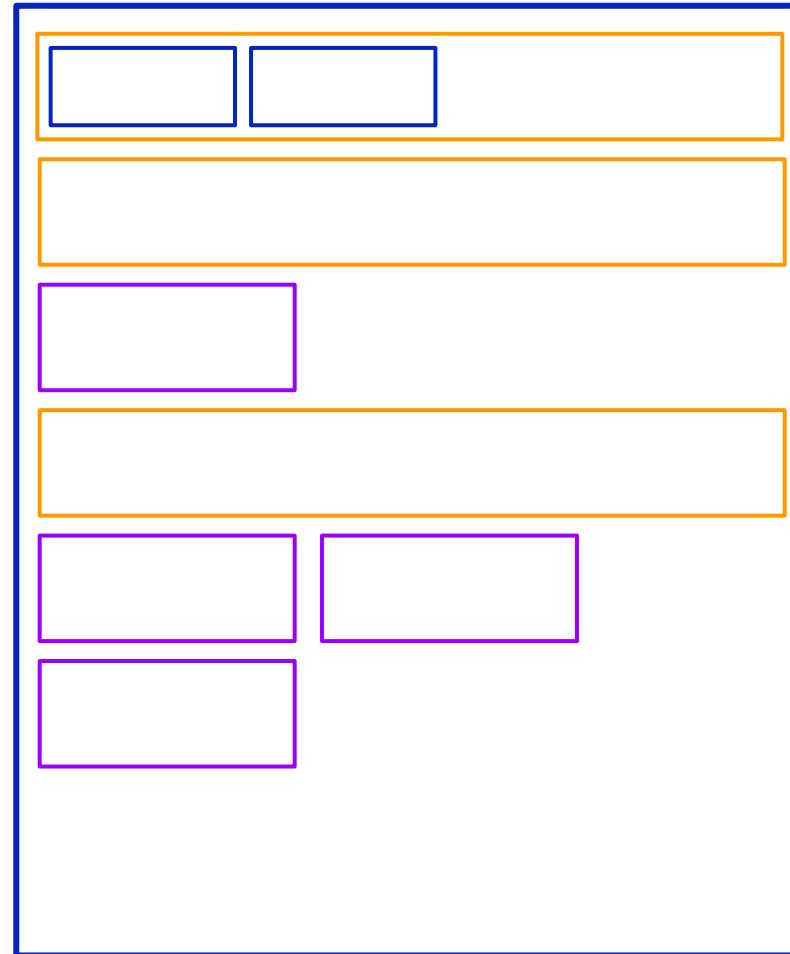
# document flow

Normal flow



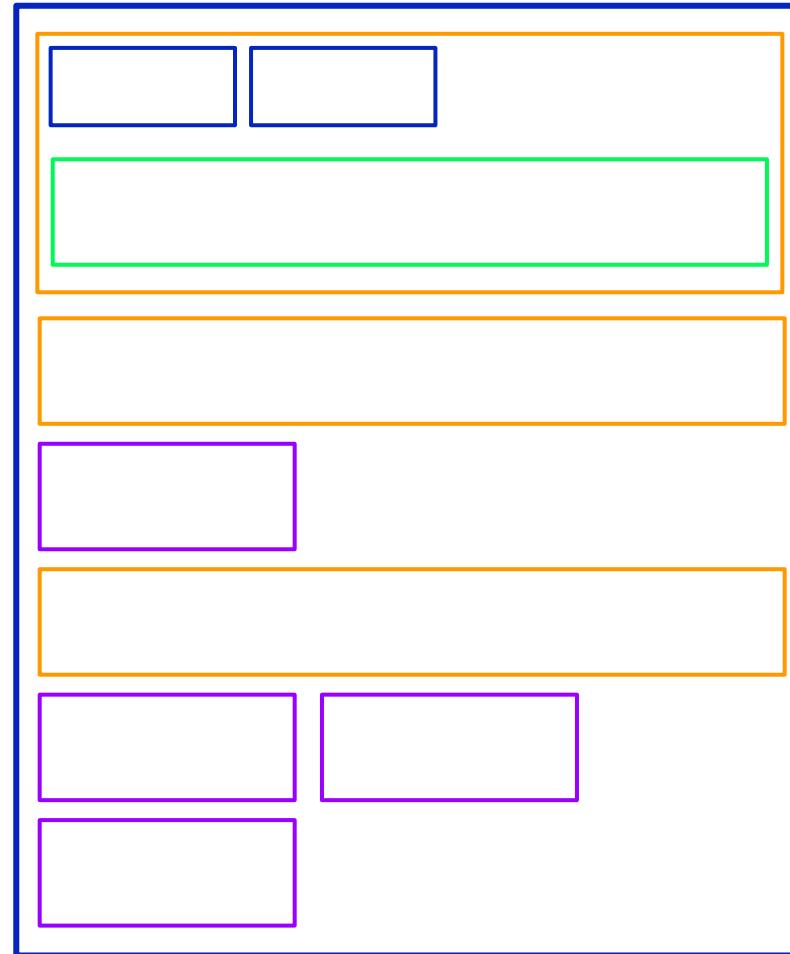
# document flow

Normal flow



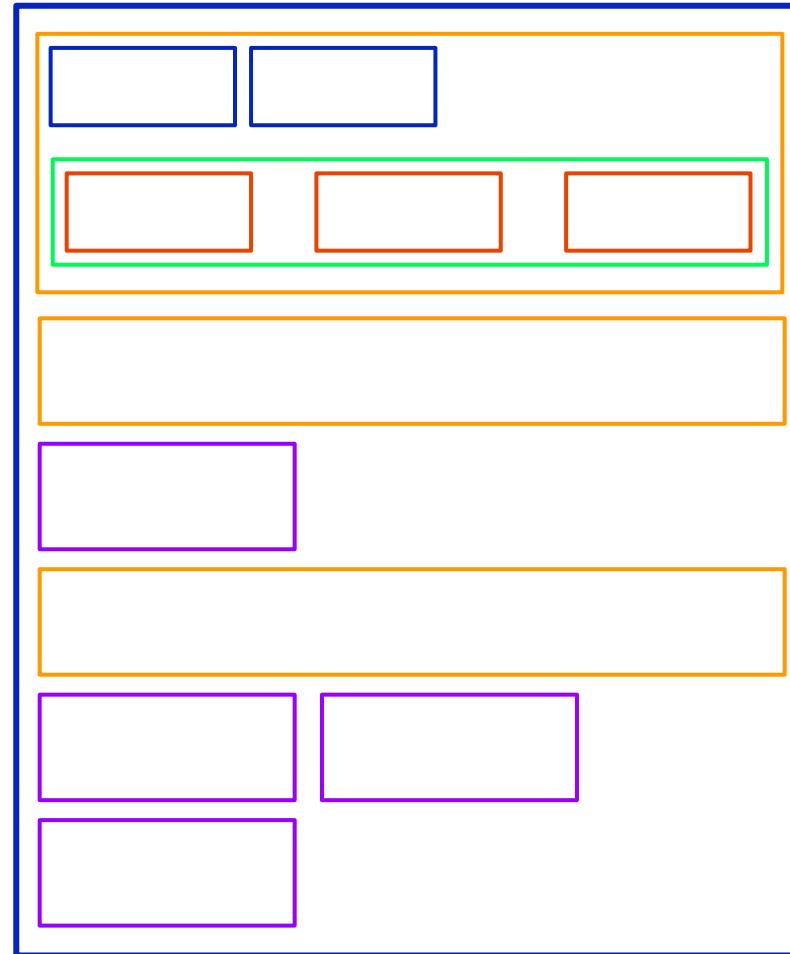
# document flow

Normal flow



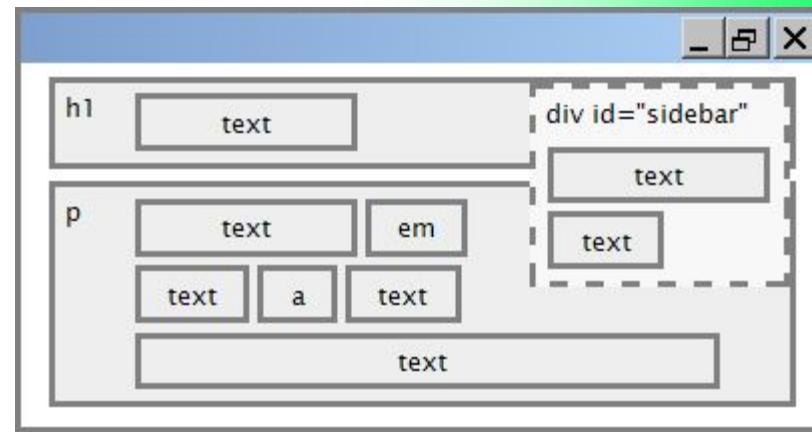
# document flow

Normal flow



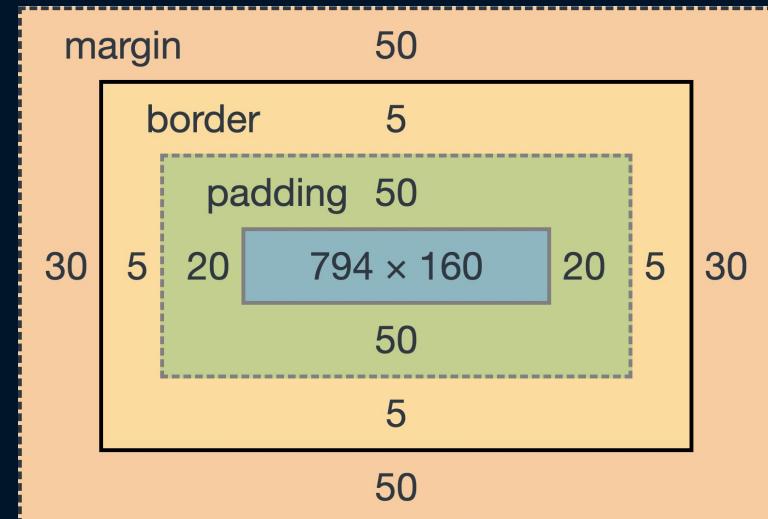
# Properties Affecting Document Flow

- display
- width, height
- position
- float
- gap, column-gap, row-gap
- margin, padding, border



# Box model

- **content**
- **padding** – The space **inside** the element, adding space **within** the element
- **border**
- **margin** – The space **outside** the element, pushing neighboring elements away.

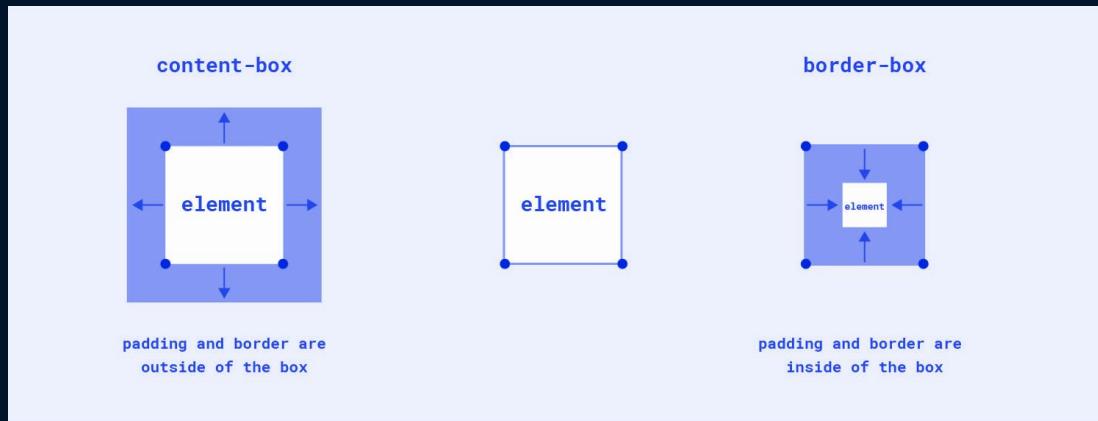


# box-sizing property

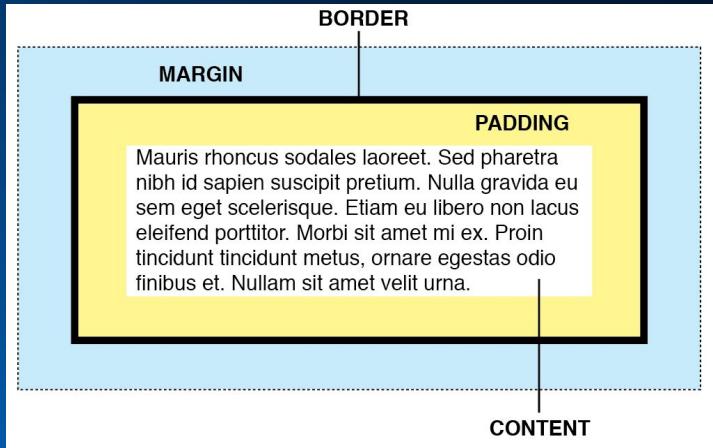
Default:

**content-box** – width and height properties include the **content**, but **does not include** the **padding**, **border**, or **margin**.

**border-box** – width and height properties **include** the **content**, **padding**, and **border**, but **do not include** the **margin**.

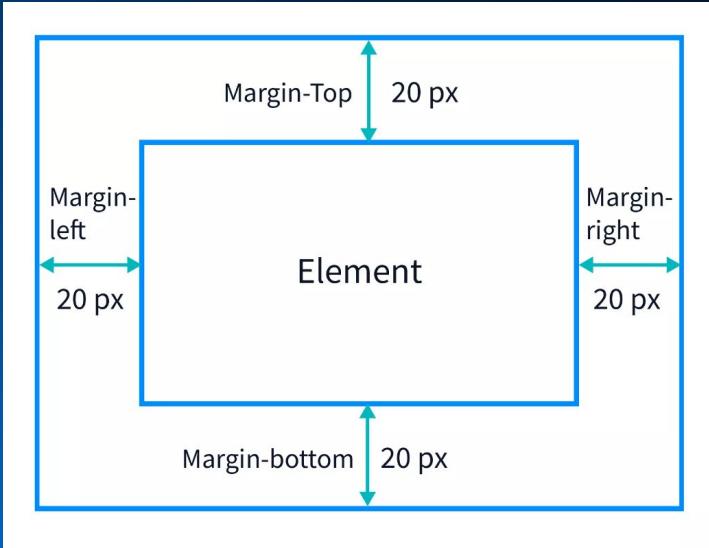


# padding

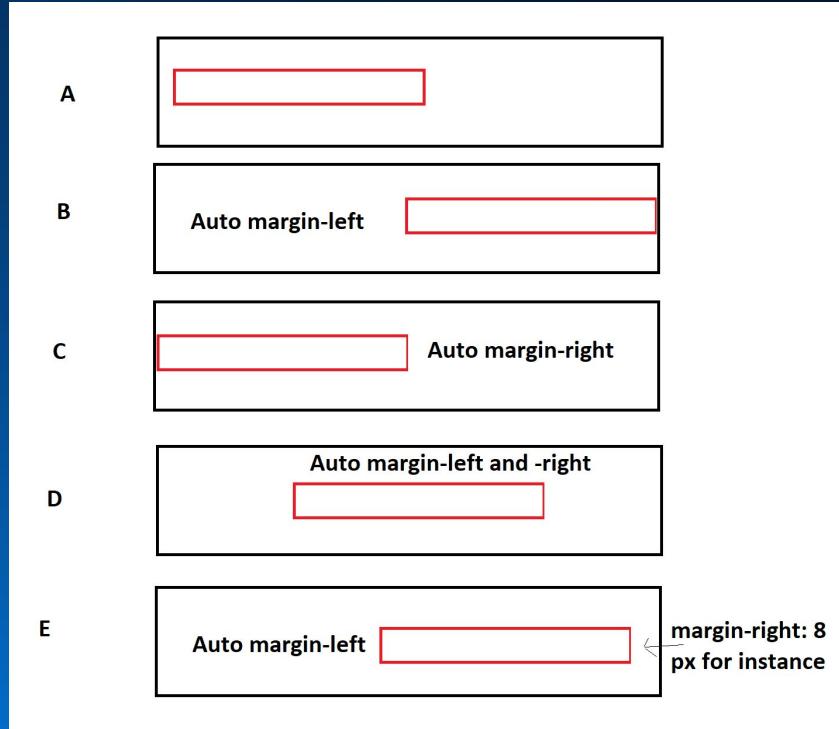


- is the internal space from the content to the edges of the element.
- Units of measurement:
  - **px**
  - **%**
- Cannot be negative.

# margin



- The outer space that pushes neighboring elements away.
- Units of measurement:
  - **px**
  - **%**
  - **auto**
- Can have negative values:  
`margin-top: -10px;`<sup>1</sup>
- `margin-top` and `margin-bottom` do not work on inline-box<sup>1</sup>



## margin: auto

`margin: auto;`  
`margin-right: auto;`  
`margin: 0 auto;`

The width of the margin is **automatically calculated** by the browser and occupies **all available space** between elements and margins within the container.

# CSS Logical Properties

Margins/padding could be applied consistently **according to the document flow**:

- use `-block` for **vertical spacing**
- use `-inline` as for **horizontal spacing**

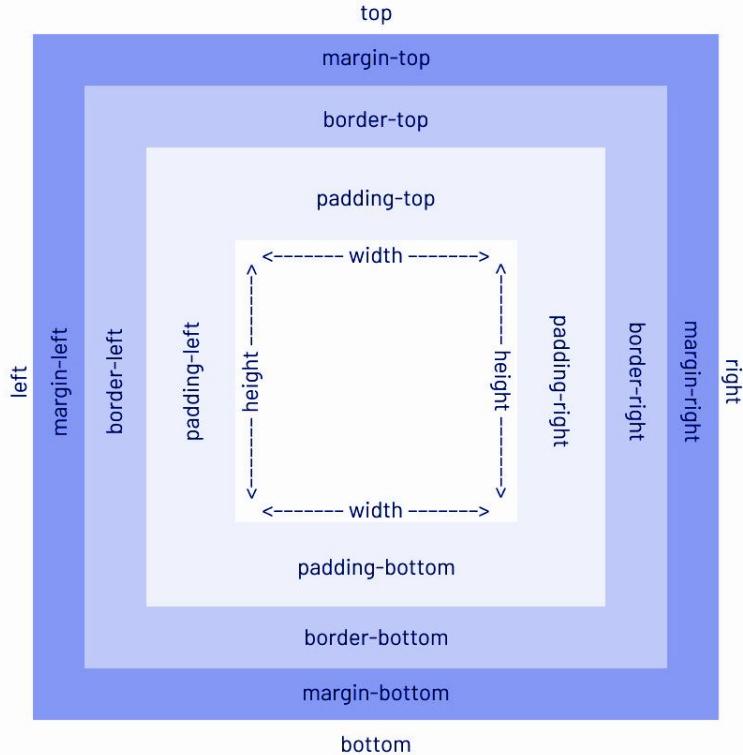
`margin-block` (`margin-top + margin-bottom`)

`margin-inline` (`margin-left + margin-right`)

`padding-block` (`padding-top + padding-bottom`)

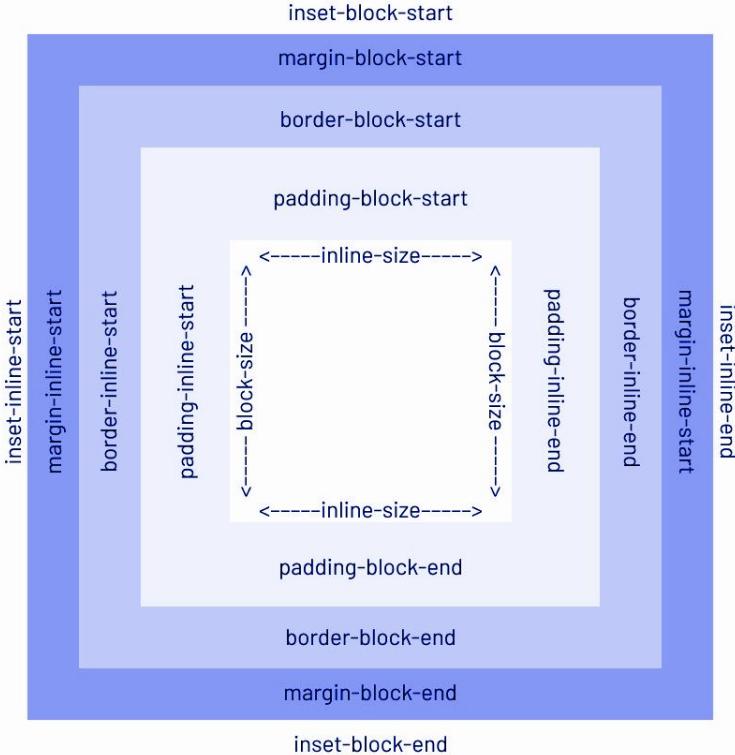
`padding-inline` (`padding-left + padding-right`)

# physical

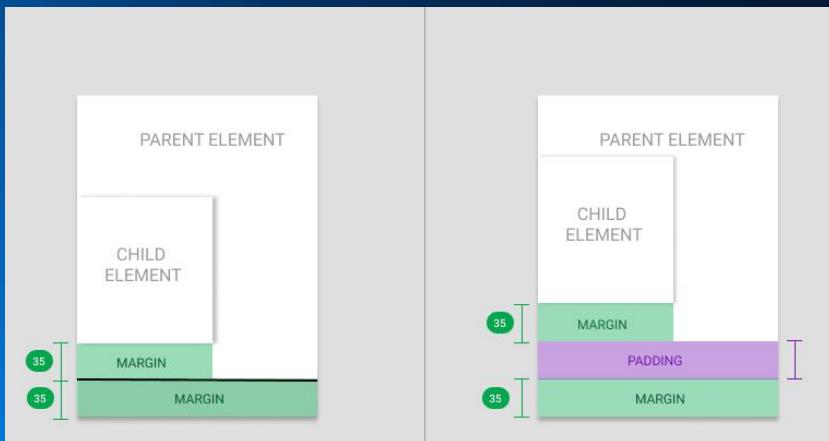
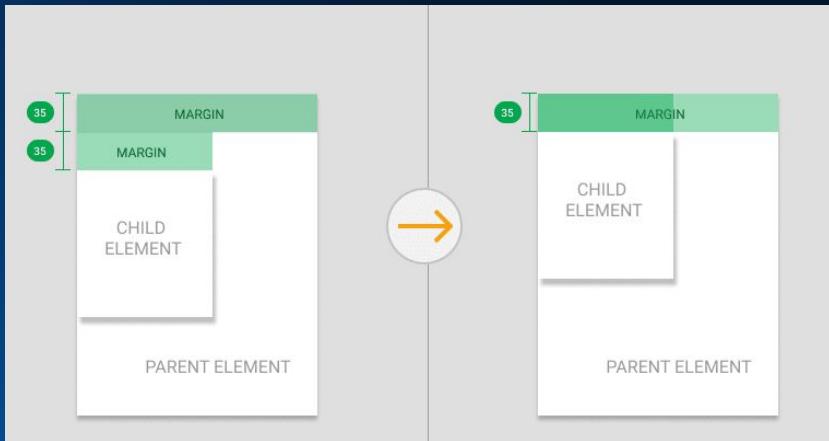


/

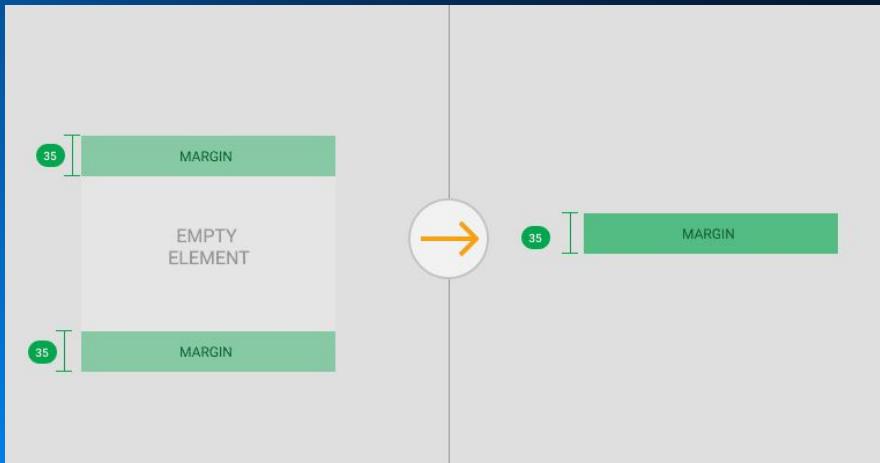
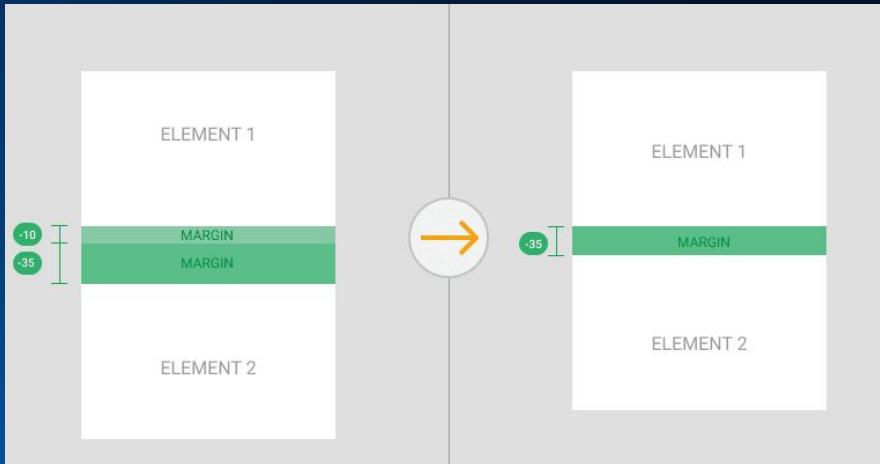
# logical



# Collapsing Margins



- Works for the **margin-top** of the first child inside a parent and for the **margin-bottom** of the last child inside a parent.
- The child's **margin** becomes the parent's **margin** or the larger of the two applies.
- Does not work if the parent has **padding**
- In elements with **position: absolute** and **position: fixed**, **display: flex**, **display: grid**; **float** margins do not collapse or cause margin drop.



# Collapsing Margins

- Works only vertically.
- Instead of adding margins together, the larger margin is applied.
- In elements with `position: absolute` and `position: fixed`, `display: flex`, `display: grid`; `float`, margins do not collapse or cause margin drop.
- If an element has no content inside, the margins still remain in the document flow.

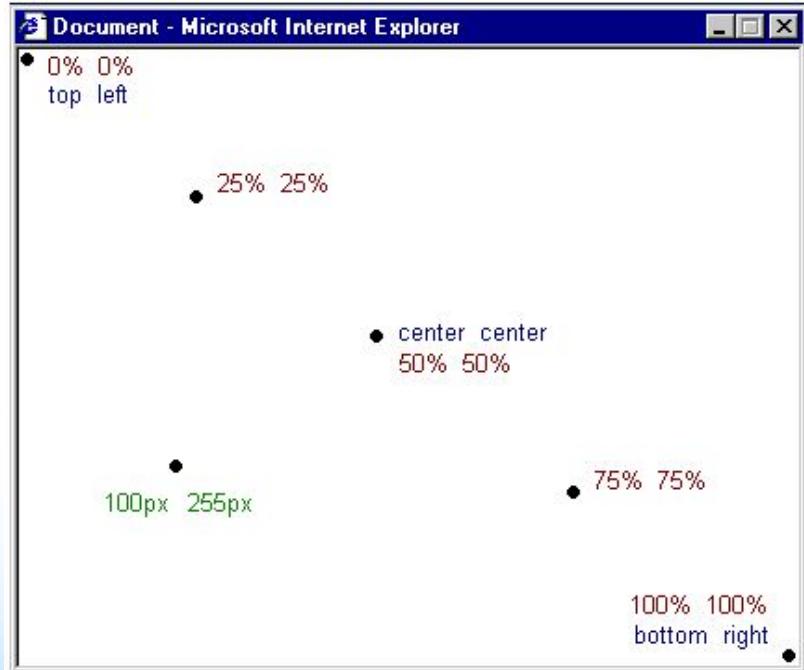
	block boxes	inline boxes	inline-block boxes
Default width	<p>Block boxes</p>	<p>Collapsed width</p>	<p>Collapsed width</p>
Default height	<p>Collapsed height</p>	<p>Collapsed height</p>	<p>Collapsed height</p>
Margins	<p>Collapsed margins</p>	<p>Margin affects sides only</p>	<p>Margin affects all sides</p>
Padding	<p>Inline boxes</p>	<p>Padding affects sides only</p>	<p>Padding affects all sides</p>
Border	<p>Border</p>	<p>Border affects sides only</p>	<p>Border affects all sides</p>

The **position** property specifies the type of positioning method used for an element:

- **static**
  - **relative**
  - **absolute**
  - **fixed**
  - **sticky**
- 
- Positioning should not be used for creating layouts, only for decorative purposes

# CSS POSITION

- **Do not use margin** to set the position of such an element.
- **top, right, bottom, left** – should **always** have at least one **horizontal and vertical** coordinate specified.
- **top, right, bottom, left** can be negative.
- You can specify all four coordinates simultaneously, but there's no guarantee that they will all work together as expected.
- **inset** – shorthand property for **top, right, bottom, left**

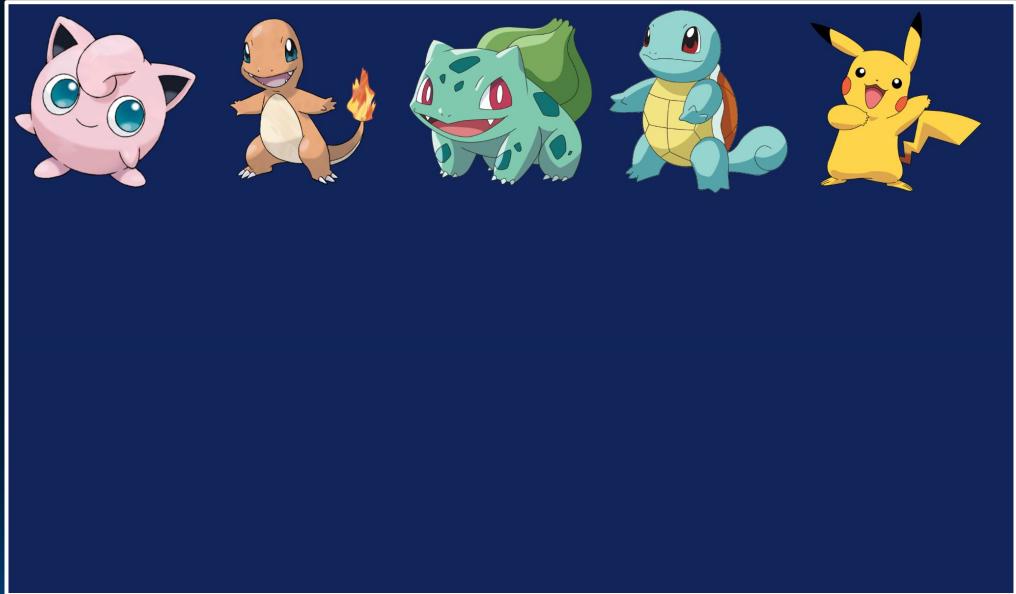


# CSS COORDINATE SYSTEM

position:

**static** [A]

- default value for all elements
- Elements take their place in the flow depending on the box type and the amount of content.
- `top, right, bottom, left, z-index` don't work
- **normal document flow**



`position: static;`

position:

## relative

- The element **remains in the normal document flow**.
- It is positioned relative to its **original position in the flow**.
- Space for the element is **still reserved** in the document flow.
- normal document flow



```
position: relative;  
left: 20px;  
top: 20px;
```

**position:**

# absolute [A]

- The element is **removed from the document flow**, and neighboring elements completely **ignore** it.
- It is positioned **relative to its parent block**  
or  
the nearest ancestor with **position: relative**.
- If both horizontal or both vertical coordinates are specified, the element will stretch between them.

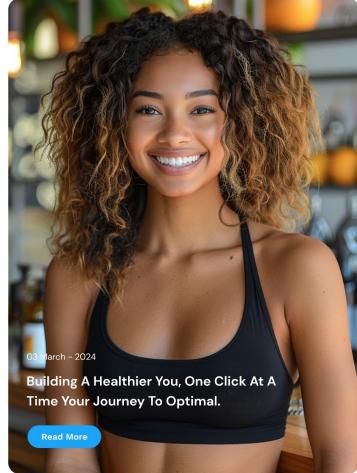


```
position: absolute;  
left: 0;  
top: 10px;
```



A screenshot of a website for "ProVice". The header includes a navigation bar with "Home", "About", "Services", "Contact", and a search icon. The main section features a woman with glasses and the text "Find the best solution together.". It includes a callout box stating "35K+ Cases Solved with Satisfaction". Below this, there's a section titled "Our Service:" with icons for "Tech Development", "Customer Insights", "Financial Planning", and "Legal Documentation". Buttons for "Get Started" and "Learn More" are also present.

A screenshot of the MOBIRISE website builder interface. At the top, there are navigation links: "OVERVIEW", "FEATURES ▾", "HELP ▾", and "DOWNLOAD". A dropdown menu under "FEATURES" is open, showing options like "Mobile friendly", "Based on Bootstrap", "Trendy blocks ▾", and "Host anywhere". To the right of the menu is a preview area showing a sunset sky background with the word "MOBIRISE" overlaid.



A screenshot of a community engagement section. The background is a gradient from green to blue. The text "JOIN OUR COMMUNITY" is prominently displayed in white. Below it, a placeholder text "Lorem Ipsum is simply dummy text of the printing and typesetting industry." is shown. At the bottom, there are two buttons: "DISCORD" with a discord icon and "WHITELIST NOW" in a green box.

position:

fixed [A]

- The element is removed from the document flow, and neighboring elements completely ignore it.
- It is positioned relative to the **viewport**.
- It does not move when **the page is scrolled**.
- If both horizontal or both vertical coordinates are specified, the element will stretch between them.

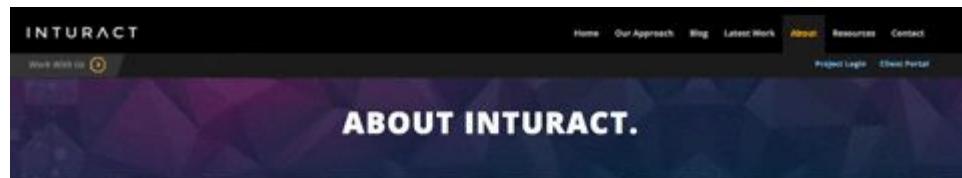
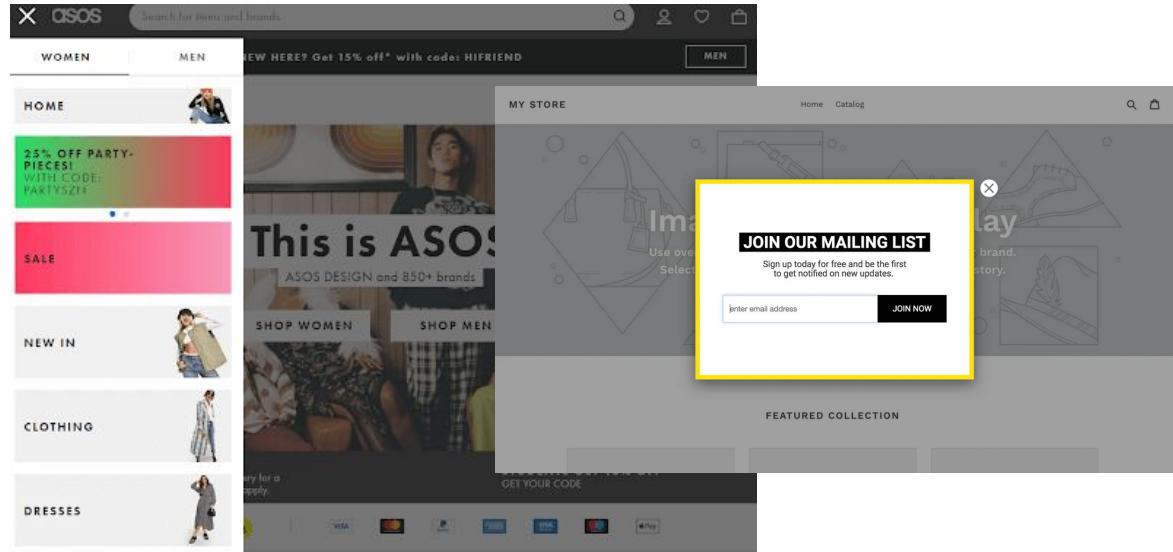


```
position: fixed;  
left: 20px;  
top: 20px;
```

## Fixed Mobile Sticky Menu

Use HTML and CSS to create a cross browser mobile navigation that sticks

By: John McGarrah



search



filters



map



radius

position:

**sticky** 

- The element is removed from the document flow.
- Neighboring elements behave as if the element is still in its original position, and space for the element is **still reserved** in the document flow.
- The element does not move when its containing **block is scrolled**.
- It behaves like **position: relative** until the block is scrolled, after which it behaves like **position: fixed**



```
position: sticky;  
right: 0;  
top: 10px;
```

**Categories**

- Fashion (1)
- Feeding (5)
- Health & Safety (1)
- Nursery (1)
- On the go (5)
- Toys (3)

**Filter by Color**

- Blue (2)
- Green (2)
- Red (1)

**Filter by Size**

- Large (1)
- Medium (1)
- Small (1)

**Filter by price**

- Filter
- Price: EGP15 — EGP25

Showing all 5 results

Sort by popularity

Long Sleeve Tee **25 EGP**

Polo **20 EGP**

T-Shirt **18 EGP**

## MAIN HEADER

Category 1

Category 2

Category 3

Settings

Main Content

Main Content

Main Content



## Build a sticky aside menu.



This is the article intro text

## Search

## Other Articles

**10 Web Design Blogs You Can't Miss**

November 8, 2017

**20 Myths About Web Design**

November 8, 2017

**14 Common Misconceptions About Web Design**

November 8, 2017

## Demo

# center position



Adds an offset upward by half the height of the block using:  
**`transform: translateY(-50%);`**

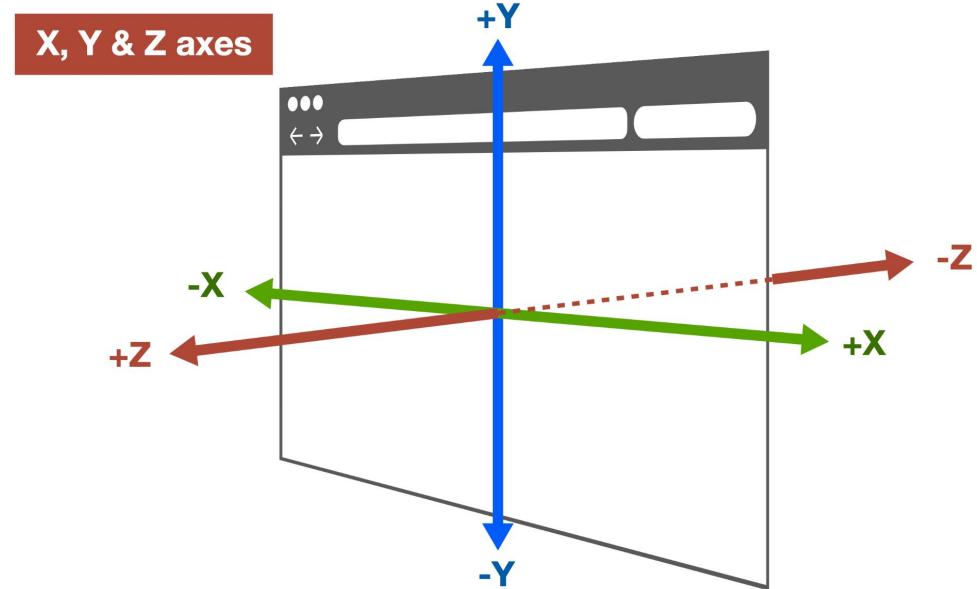
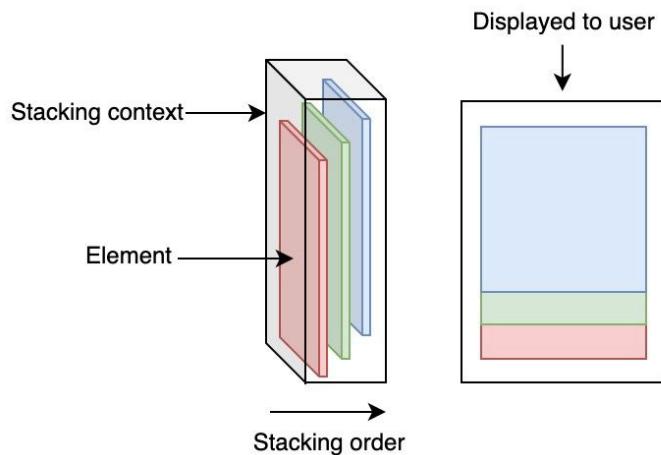
Adds an offset to the left by half the width of the block using:  
**`transform: translateX(-50%);`**

or

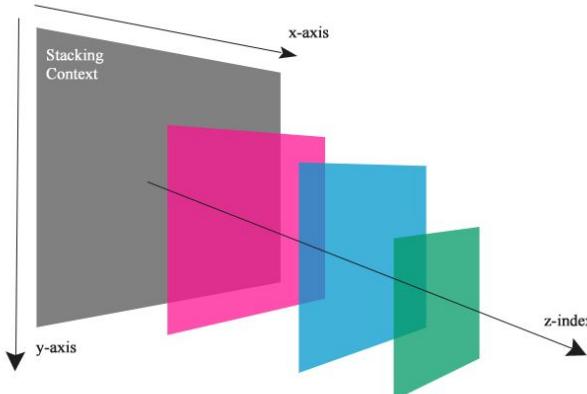
**`translate: -50% -50%;`**

# stacking context

x – left / right  
y – top / bottom  
z – z-index



# z-index

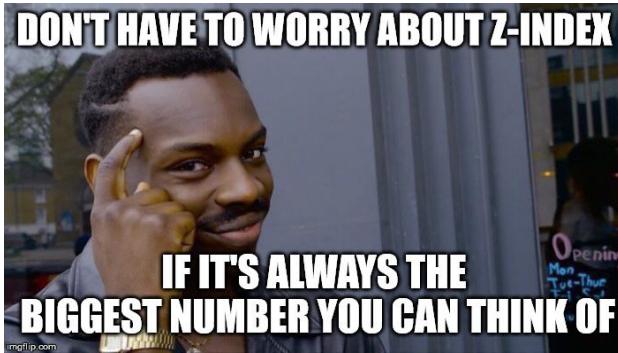


- Controls the stacking order of elements within a stacking context.
- Applies only to elements with an explicitly defined `position` other than `static`.
- If positioning is not needed but you still require a z-index, use `position: relative` without coordinates.
- Values can be positive or negative numbers.
- The higher the number, the more visible the layer.
- By default, `z-index: auto` is used, and visibility is determined by the element's position in the DOM tree: the lower it is in the code, the more visible it is.

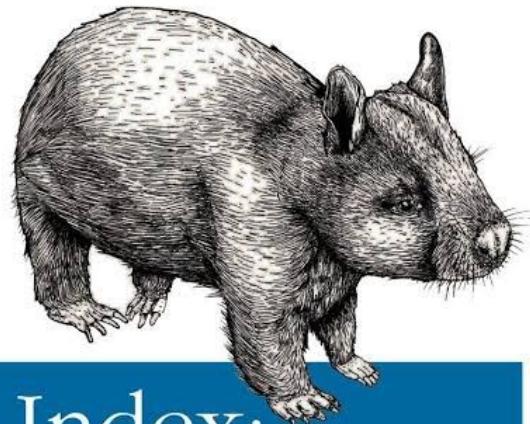
# z-index

- Stacking context determines the order of how elements are layered on top of each other

Z-index should start from 1 and increment by +1 to avoid using random large numbers like 10, 83, 100, etc.



*You've come this far, no going back now.*



*Real World CSS*

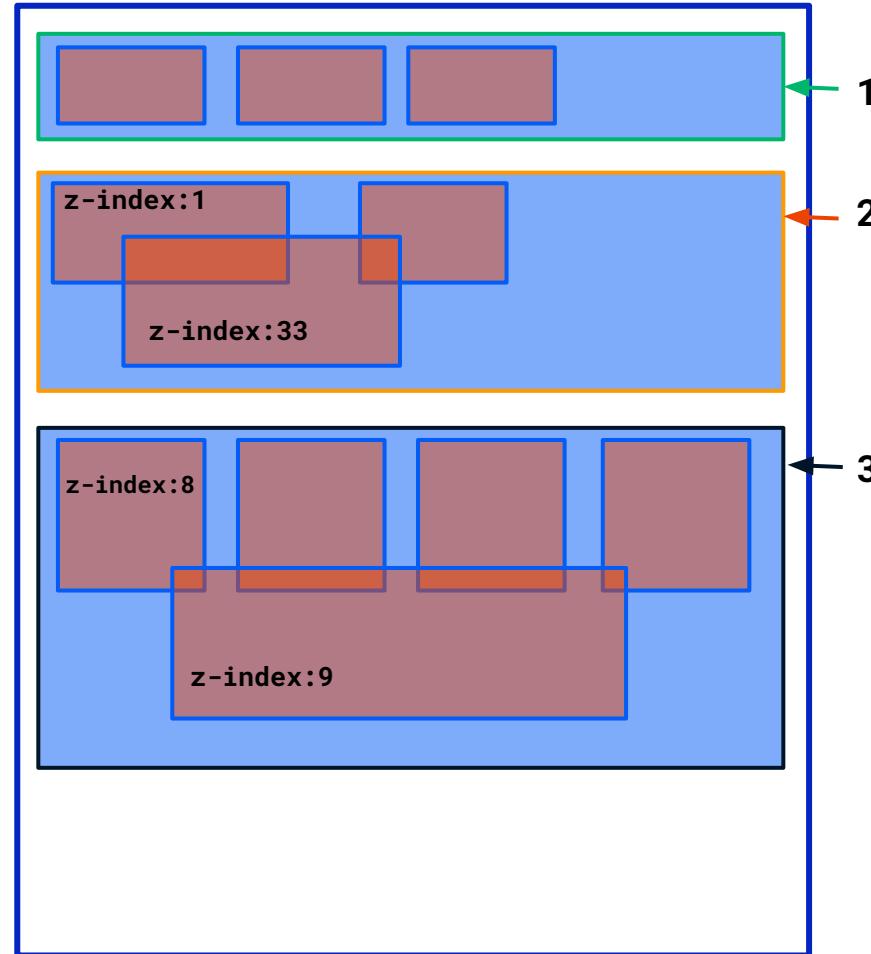
O RLY?

*@ThePracticalDev*

## stacking<sup>[A]</sup>

# context

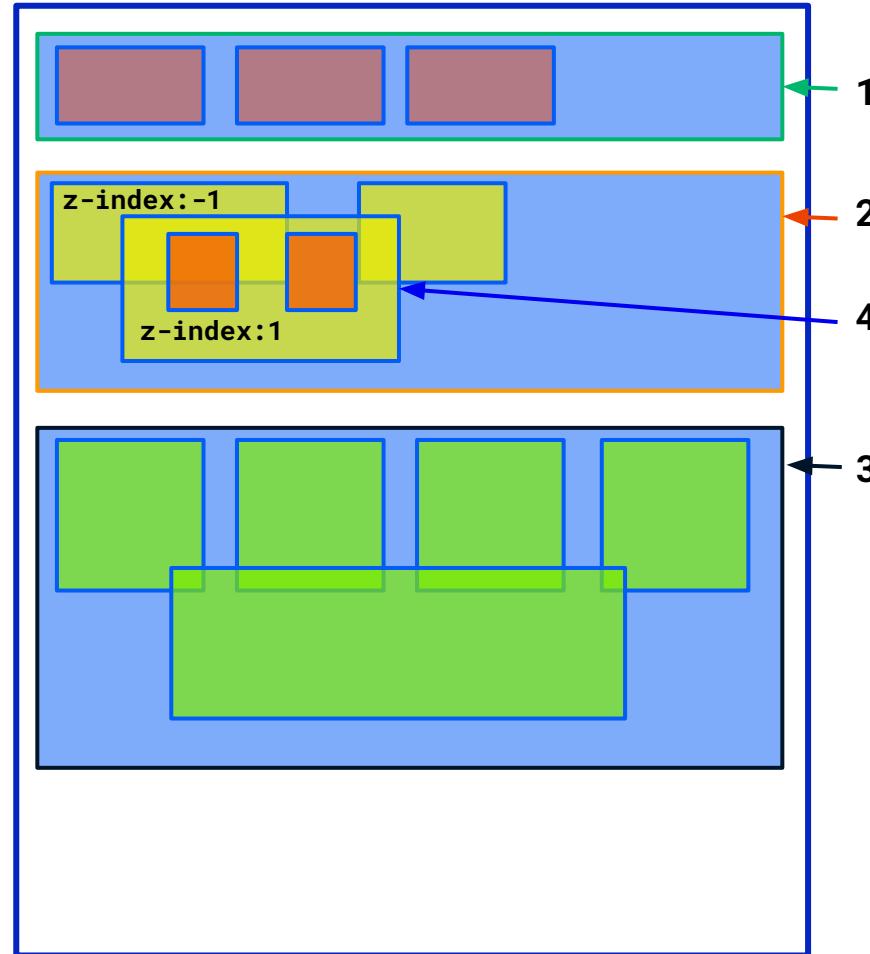
- There can be multiple independent stacking contexts on a page.
- **z-index** property applies to child elements within an element that creates a stacking context.



## stacking<sup>[IA]</sup>

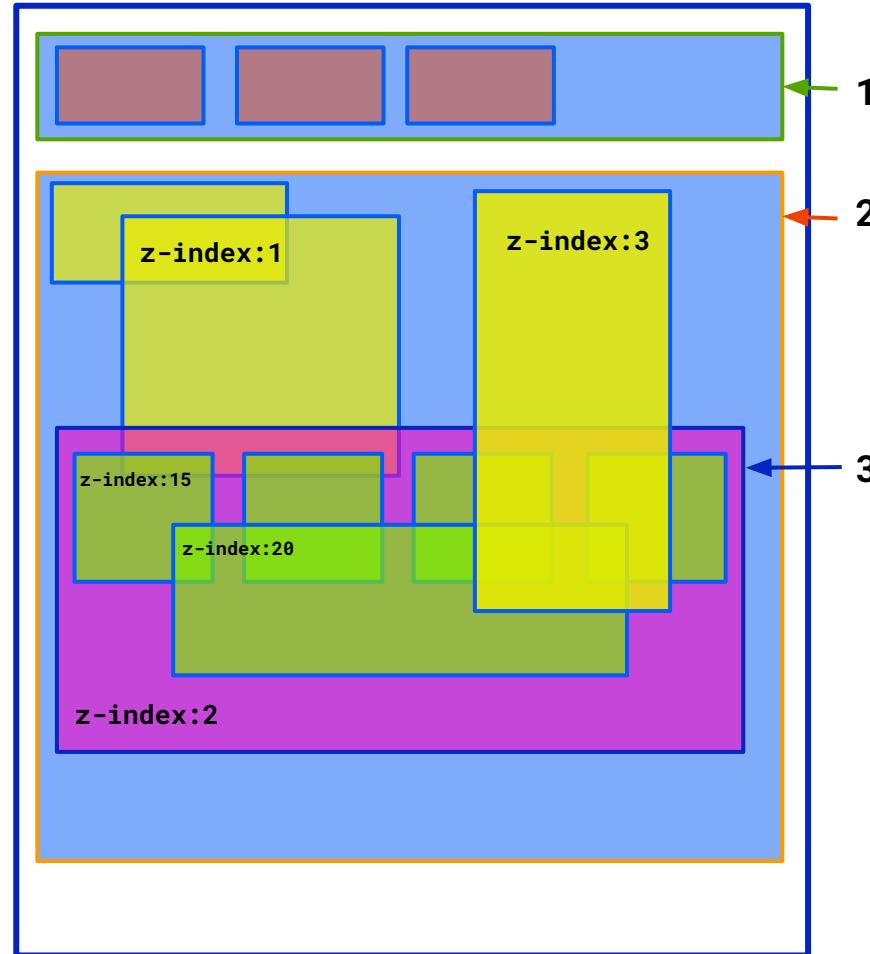
# context

Elements that are direct descendants of the same parent belong to the same stacking context.



# stacking context

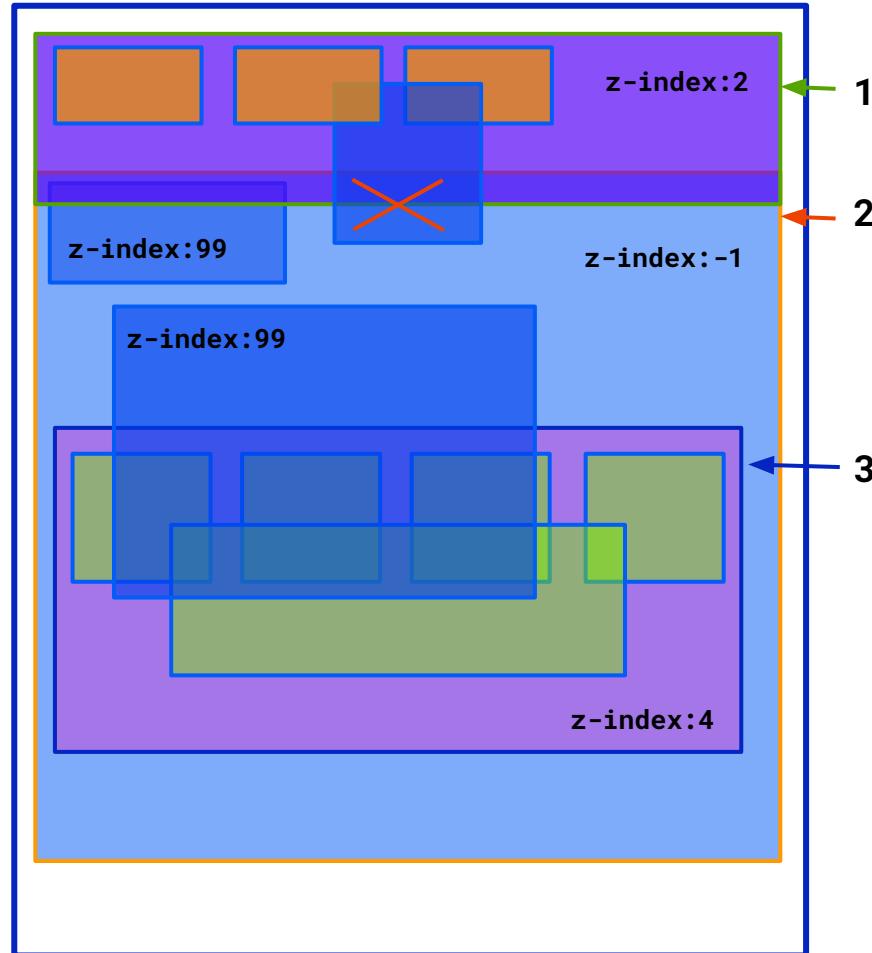
- One stacking context can be nested within another.
- Child elements are considered part of their parent's stacking context.



## stacking<sup>[IA]</sup>

# context

Elements within one stacking context cannot be positioned between elements of another stacking context.



# stacking context

**root stacking context** – `<html>` tag creates the root stacking context, unless other conditions are specified:

- `position: absolute` with a `z-index` other than `auto`
- `position: relative` with a `z-index` other than `auto`
- `position: fixed`
- `position: sticky` (only on mobile devices)
- A child element of a `flex` container with a `z-index` other than `auto`
- `opacity` less than 1
- `transform` with a value other than `none`
- `filter` with a value other than `none`
- `perspective` with a value other than `none` or 0

# float

allows content to flow around a block element.

The block is pushed to the specified side as far as possible.

float document flow [A]

**float: left;**  
**: right;**  
**: both;**  
**: none;**

Do not use float for building layouts!



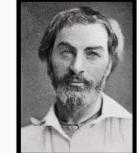
Lorem ipsum dolor sit amet, consectetur adipisciing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Lorem ipsum dolor sit amet, consectetur adipisciing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

“ *J*his is what you shall do; Loose the earth and sun and the animals, despise riches, give alms to every one that asks, stand up for the stupid and crazy, devote your income and labor to others, hate tyrants, argue not concerning God, have patience and indulgence toward the people, take off your hat to nothing known or unknown or to any man or number of men, go freely with powerful uneducated persons and with the young and with the mothers of families, read these leaves in the



Photo by Jeremy Bishop  
on Unsplash



Walt Whitman

open air every season of every year of your life, re-examine all you have been told at school or church or in any book, dismiss whatever insults your own soul, and your very flesh shall be a great poem and have the richest fluency not only in its words but in the silent lines of its lips and face and between the lashes of your eyes and in every motion and joint of your body.” – Walt Whitman, Song of Myself

Demo

# clear float

is used to stop elements from being wrapped by floated elements.

It can be applied to:

1. An element that follows a floated element - **float**

**clear: left;**  
**: right;**  
**: both;**  
**: none;**



Lore ipsum dolor sit amet, con sectetur adip iscing elit, sed do eius mod tempor in cididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exer citat ion ullamco laboris nisi ut aliquip ex ea com modo consequat. Duis aute irure dolor in repre henderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Lore ipsum dolor sit amet, con sectetur adip iscing elit, sed do eius mod tempor in cididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exer citat ion ullamco laboris nisi ut aliquip ex ea com modo consequat. Duis aute irure dolor in repre henderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.



**Demo**

# clear float

It can be applied to:

2. An element that follows a floated element – **float**

**Alternatives:**

**overflow:** `hidden; `

**display:** `flow-root; `



*Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.*

*Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.*

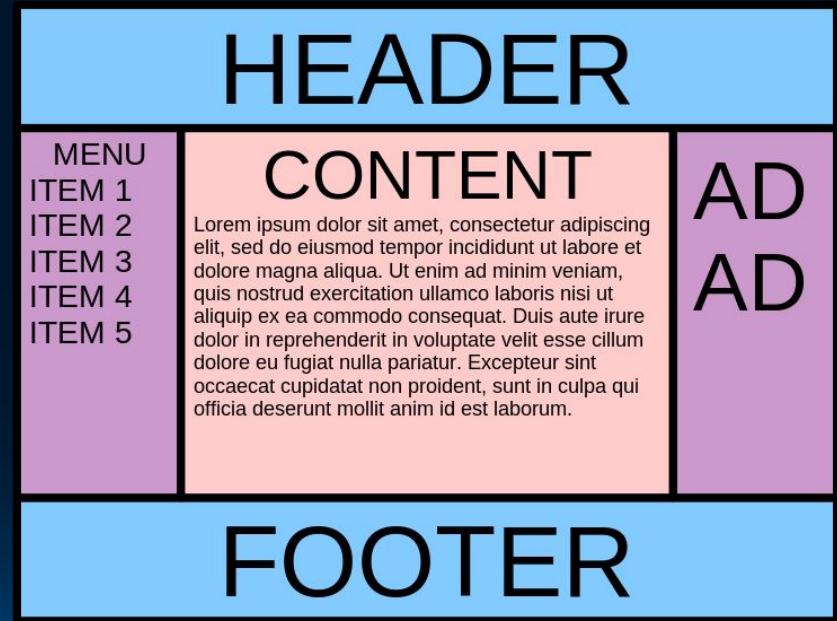
**Demo**

# flexbox

Flexbox allows control over:

- Size
- Order
- Alignment of elements across multiple axes
- Distribution of free space between elements
- and more

- ★ Ease in creating complex layouts
- ★ Automatic alignment
- ★ Responsiveness



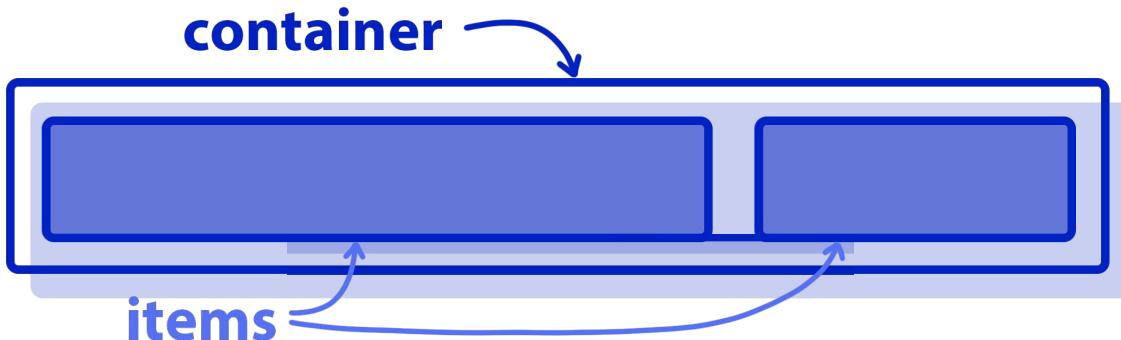
# flex container and items

The container is given the property:

`display: flex;`

Style properties are applied to the **elements** inside the container.

The elements adjust to the width of their content and are arranged in a row by default.



[Demo](#)

# flex: axis

There are two axes:

**main axis horizontal**

**cross axis vertical**

The direction of the axes can be changed using the property:

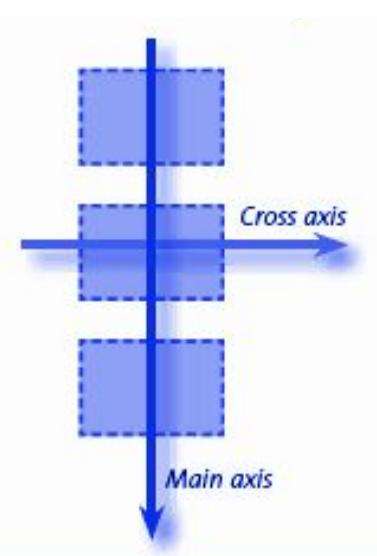
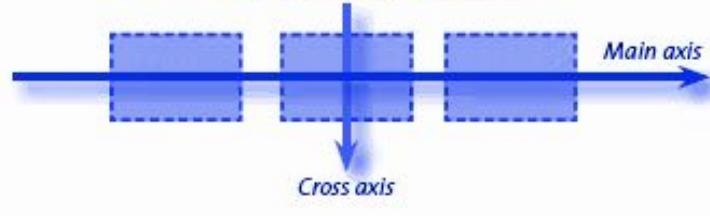
**flex-direction**

Default:

**flex-direction: row;**

To arrange elements in a column:

**flex-direction: column;**



**Demo**

# flex:

# Changing the Order of Elements

[Demo](#)

1. Using Reverse Direction:

```
flex-direction: row-reversed;  
flex-direction: column-reversed;
```

---

2. The **order** Property:

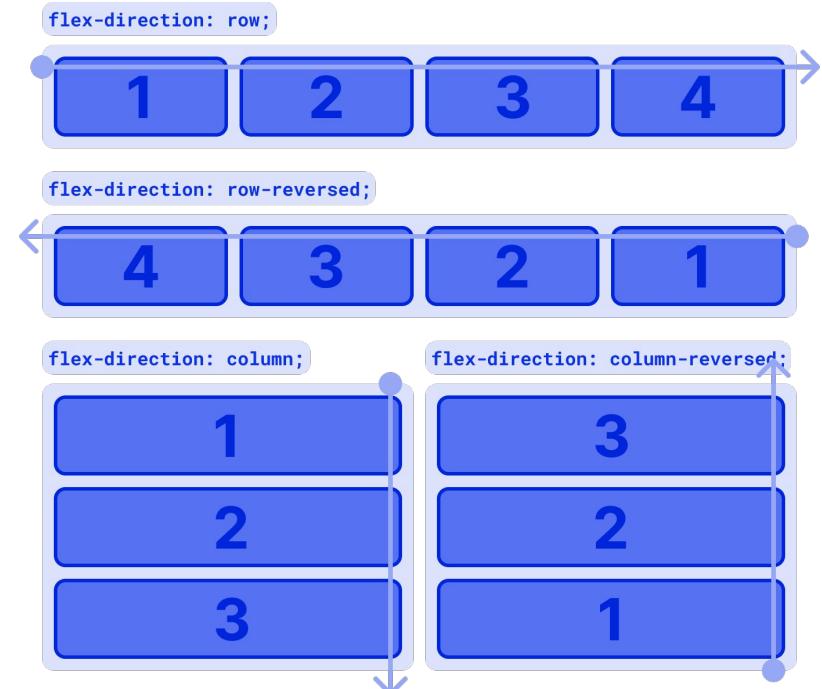
```
order: 4;
```

Accepts positive and negative values.

Works correctly when the **order** property is set for all elements within the flex container.



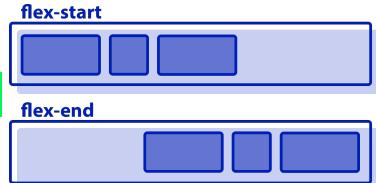
```
-1 1 1 5 99
```



# flex: Alignment elements

Main Axis:

**justify-content** ↗



Default Value:

**flex-start** ↗

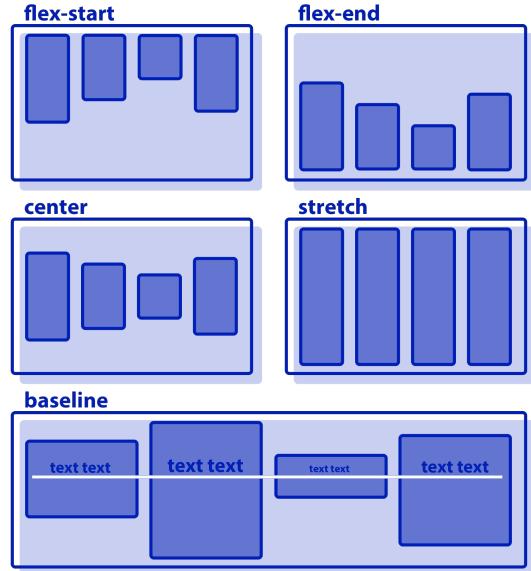
Demo

Cross Axis:

**align-items** ↗

Default Value:

**stretch** ↗



Demo

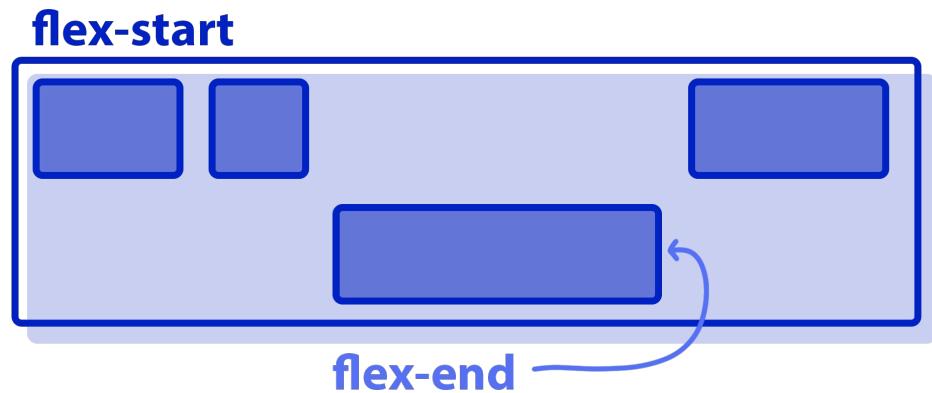
These properties are set on the container, and the alignment is applied to the items within it.

# **flex:** Alignment one element

You can align a specific element only along the cross axis using:

**align-self** [A]

The values are the same as those for  
**align-items**

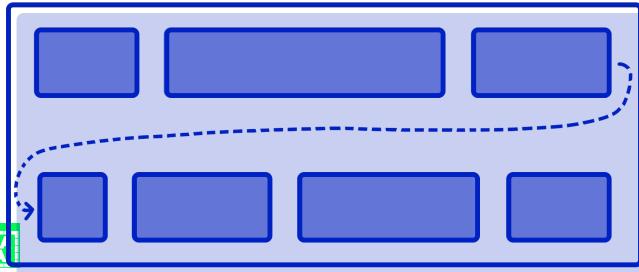


[Demo](#)

# flex: Working with Rows

Flexbox can wrap items to new rows using :

`flex-wrap: wrap;`



Default:

`flex-wrap: no-wrap;`

Aligning Rows on the Cross Axis:

`align-content`

Demo

Default:

`align-content: flex-start;`

Demo

`align-content`

`flex-start`

`flex-end`

`center`

`stretch`



`space-between`

`space-around`



# flex: Spacing Between Items

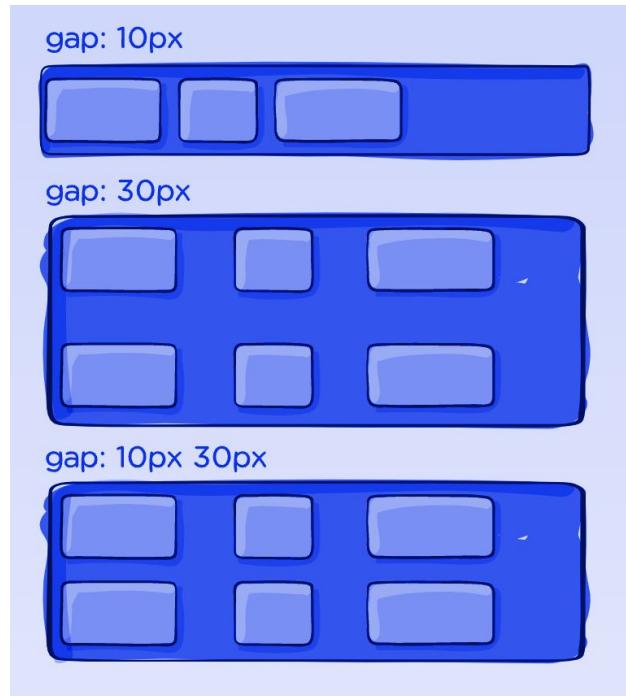
You can set **margin** for each element individually, but it's much easier to set spacing for the entire container in one place using the **flex** container.

**gap** – Sets spacing between items

**gap** : 10px 15px;

**row-gap** : 15px;

**column-gap** : 10px ;



# flex: distributing extra space

Applied along the **main axis** for **flex-item**.

Coefficient for distributing extra space:

**flex-grow** 

Values are positive integers greater than 0.

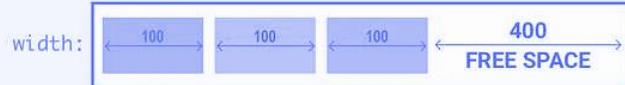
**flex-grow: 2; 1; 0** 

The distribution is calculated based on a specific formula ->

## Demo

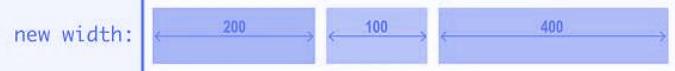
### flex-grow calculation

$$\text{new width} = \left( \frac{\text{flex grow}}{\text{total flex grow}} \times \text{free space} \right) + \text{width}$$



flex-grow: **1      0      3**

$$\text{calculation: } \left( \frac{1}{4} \times 400 \right) + 100 \quad \left( \frac{0}{4} \times 400 \right) + 100 \quad \left( \frac{3}{4} \times 400 \right) + 100$$



# flex: shrinking elements

Applied along the **main axis**.

Shrink coefficient when items don't fit in one line:  
**flex-shrink**

Values are positive integers greater than 0.

**flex-shrink: 1;**

The shrinkage is calculated based on a specific formula →

**flex-shrink: 0;** – The element will maintain the width specified by **width** or **flex-basis**, even if this causes other elements to overflow.

## flex-shrink calculation

new width = width – (shrunk space × shrink ratio)



flex-shrink: 4 6

calculation: total shrink scaled width =  $\sum (\text{width} \times \text{flex shrink})$   
 $(300 \times 4) + (600 \times 6) = 4800$

$$\text{shrink ratio} = (\text{width} \times \text{flex shrink}) / \text{total shrink scaled width}$$
$$(300 \times 4) / 4800 = 0.25 \quad (600 \times 6) / 4800 = 0.75$$

$$\text{new width} = \text{width} - (\text{shrunk space} \times \text{shrink ratio})$$
$$300 - (100 \times 0.25) = 275 \quad 600 - (100 \times 0.75) = 525$$

new width: 275 525

# flex: base size of an element

**flex-basis** defines the size of an element along the **main axis** before considering **flex-grow** and **flex-shrink**.

Values can be in **px** or **%**.

**flex-basis: 50%;**

Child	flex-basis
	flex-basis: auto flex-basis: auto
	flex-basis: 50% flex-basis: 50%
	flex-basis: auto flex-basis: 500px

flex-basis vs widths	
<pre>... .child {   flex-basis: 500px   width: 100px }</pre>	flex-basis wins 500px
<pre>... .child {   min-width: 100px   flex-basis: 500px }</pre>	min-width wins 100px
<pre>... .child {   max-width: 700px   flex-basis: 500px }</pre>	max-width wins 700px

# **flex:** shorthand property

flex-grow / flex-shrink / flex-basis [W]

**flex:** 1 0 300px;

Flex



Grow

Shrink

Basis

# Centering Content, Full-Width Background

The background of the website should always stretch across the full width of the page.

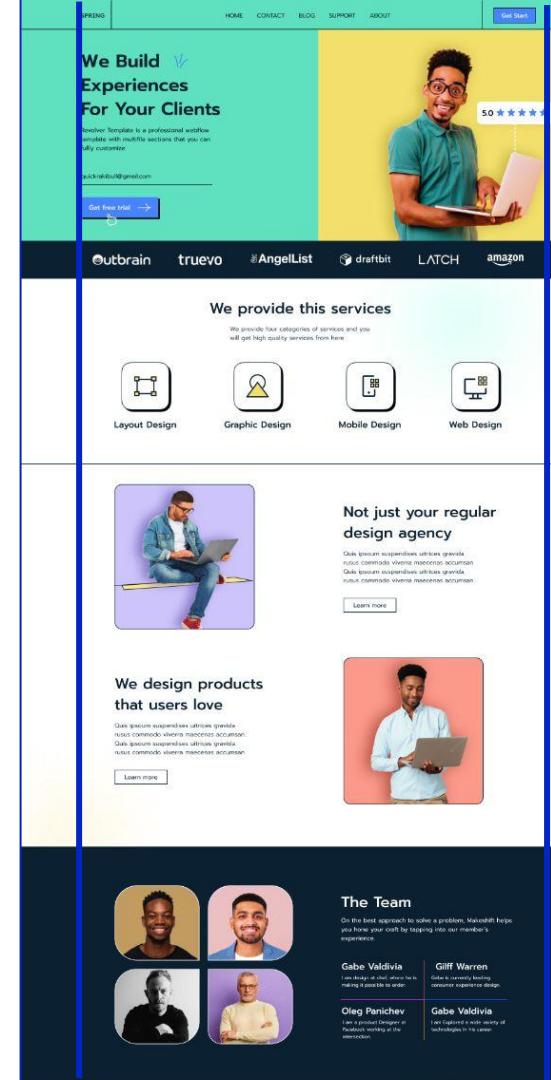
The content has a width restriction (set by width/max-width) as per the design layout and is centered on the page.

```
<section>
  <div class="content-wrapper">

    section { background-color: grey; }

    .content-wrapper {
      width: 1400px;
      margin: 0 auto;
    }
```

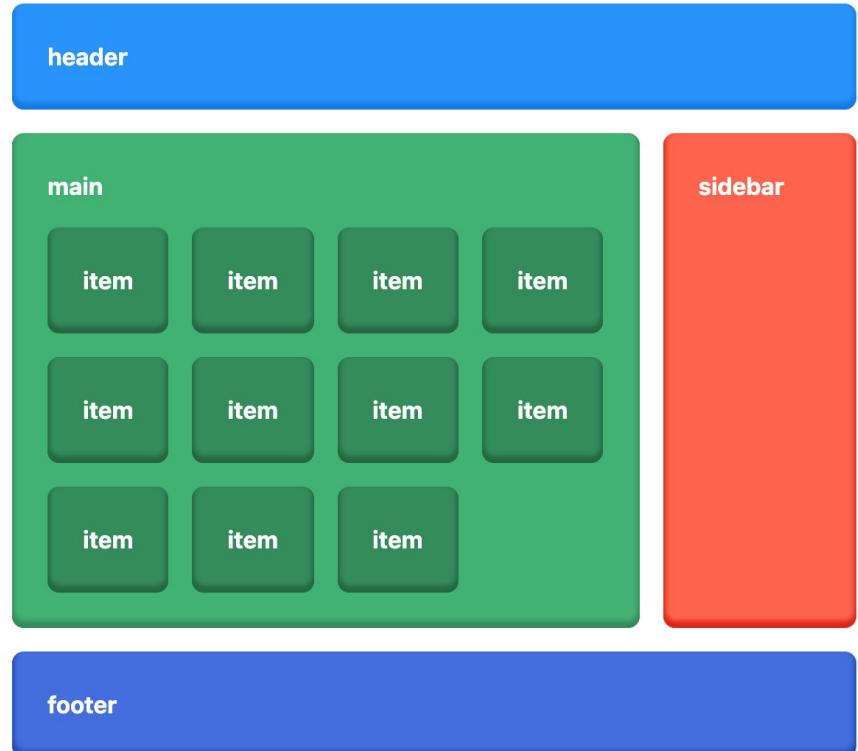
Demo



example

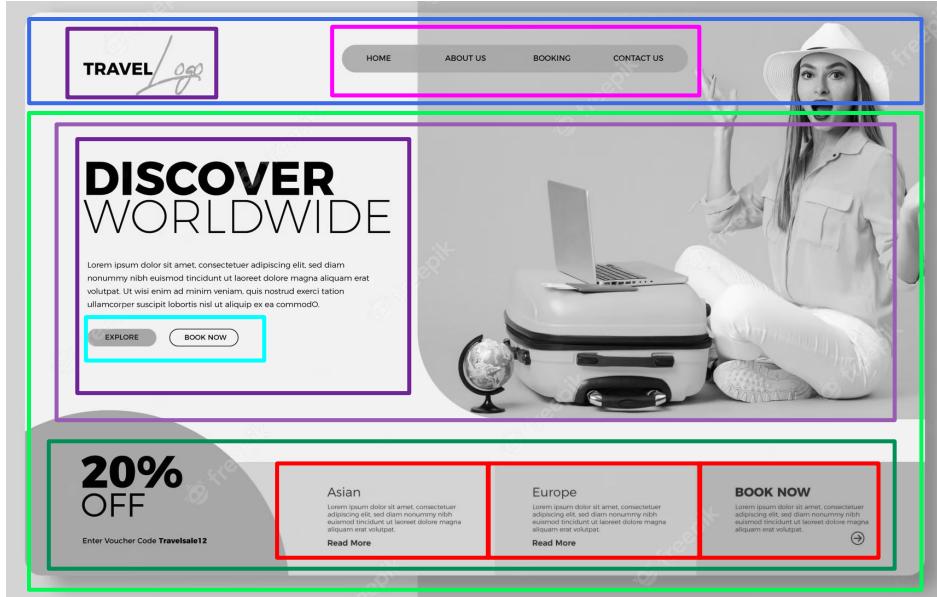
# flex golden layout

```
body {  
    display: flex;  
    flex-wrap: wrap;  
    gap: 20px;  
}  
  
header,  
footer {  
    flex-basis: 100%;  
}  
  
main {  
    flex-basis: 80%;  
    display: flex;  
    flex-wrap: wrap;  
}  
  
item {  
    flex-grow: 1;  
}  
  
sidebar {  
    flex-basis: 20%;  
}
```



flex

# hero section



```
.header {  
    display: flex;  
}  
.nav {  
    display: flex;  
    margin: 0 auto;  
}  
.main {  
    display: flex;  
    flex-direction: column;  
}  
.hero-text {  
    display: flex;  
    flex-direction: column;  
    margin-right: auto;  
}  
.button-wrap {  
    display: flex;  
}  
.advantages {  
    display: flex;  
}  
.advantages-item {  
    flex-grow: 1;  
}
```

# flex card item



.is-gallery

My Everyday Sweater  
Item No. 106358 \$108.00

★★★★★ (10)

Select a color

Select a size

XSMALL SMALL MEDIUM LARGE  
XLARGE XXLARGE

Size Guide Available in Petite Sizes

Add a Gift Box

1 ADD TO BAG

.is-info

justify-content: space-between  
justify-content: flex-start  
justify-content: space-between

```
.card {  
  display: flex;  
}  
.part {  
  flex-grow: 1;  
  display: flex;  
}  
.is-gallery {  
  flex-wrap: wrap;  
}  
.is-info {  
  flex-direction: column;  
}  
.row {  
  display: flex;  
}  
.multirow {  
  display: flex;  
  flex-wrap: wrap;  
}
```

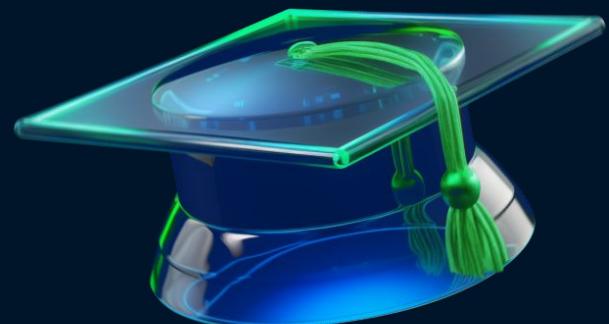
# Quality Criteria for HTML Course

- ❤️ Mandatory for passing the course
- 💛 Required for the highest grade
- 💚 Optional

❤️ 2.12 Use flex or grid for layouts  
Do not use tables or positioning.

# Summary

1. How elements are positioned in the document flow
2. What is included in the box model
3. Different types of boxes
4. Specifics of margin behavior
5. How positioning and stacking context work
6. The purpose of float
7. How to build a page layout using Flexbox



# Homework

1. Complete a Course about box model and flexbox
2. Interactive Flexbox Learning (Optional but Encouraged)
3. Create a Page Layout Using Flexbox
  - Create a page layout using Flexbox for the layout system.
  - Implement inner (padding) and outer (margin) spacing for the elements.
  - Use positioning properties (relative, absolute, fixed, etc.) where necessary, especially for pseudo-elements.





# QUESTIONS?



**Please fill out the feedback form**  
**It's very important for us**



**THANK YOU!**  
**Have a good evening!**