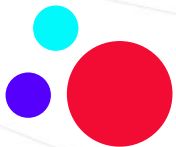


# CSS Units & Flexbox

infoShare Academy



# HELLO

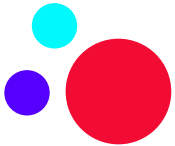
## Dawid Buliński

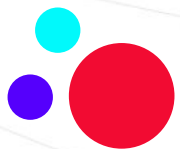
Front End Developer



# Agenda

1. Dostępne jednostki miary w CSS
2. Dokumentacja – gdzie szukać informacji?
3. Jednostki absolutne
4. Jednostki względne
5. Opis flexboxa
6. Flexbox container
7. Flexbox items





# Dostępne jednostki w CSS

## ABSOLUTE

Pixels (px)

Inches (in)

Centimeters (cm)

Millimeters (mm)

Points (pt)

Picas (pc)



## RELATIVE

Percentages (%)

Font sizes (em, rem)

Character sizes (ex, ch)

Viewport dimensions (vh, vw)

Viewport max (vmax)

Viewport min (vmin)

<https://medium.com/nerd-for-tech/everything-you-need-to-know-about-css-units-f09a94acd793>



[https://developer.mozilla.org/en-US/docs/Learn/CSS/Building\\_blocks/Values\\_and\\_units](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units)

[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Values\\_and\\_Units](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Values_and_Units)

<https://css-tricks.com/the-lengths-of-css/>



# Jednostki absolutne

Najpopularniejszą jednostką absolutną, którą spotykamy w CSS jest **pixel** (px). Pomimo tego, że jest to wartość absolutna, to warto wiedzieć px w CSS nie jest tożsamy fizycznemu pixelowi na ekranie.

Na przykład: **iPhone 12 Pro**, charakteryzuje się ekranem o rozdzielczości 2532 × 1170 px. Jednak ze względu na skalowanie (wartość **devicePixelRatio: 3**) jeden pixel jako jednostka w CSS równa jest 3 rzeczywistym pixelom na wyświetlaczu.



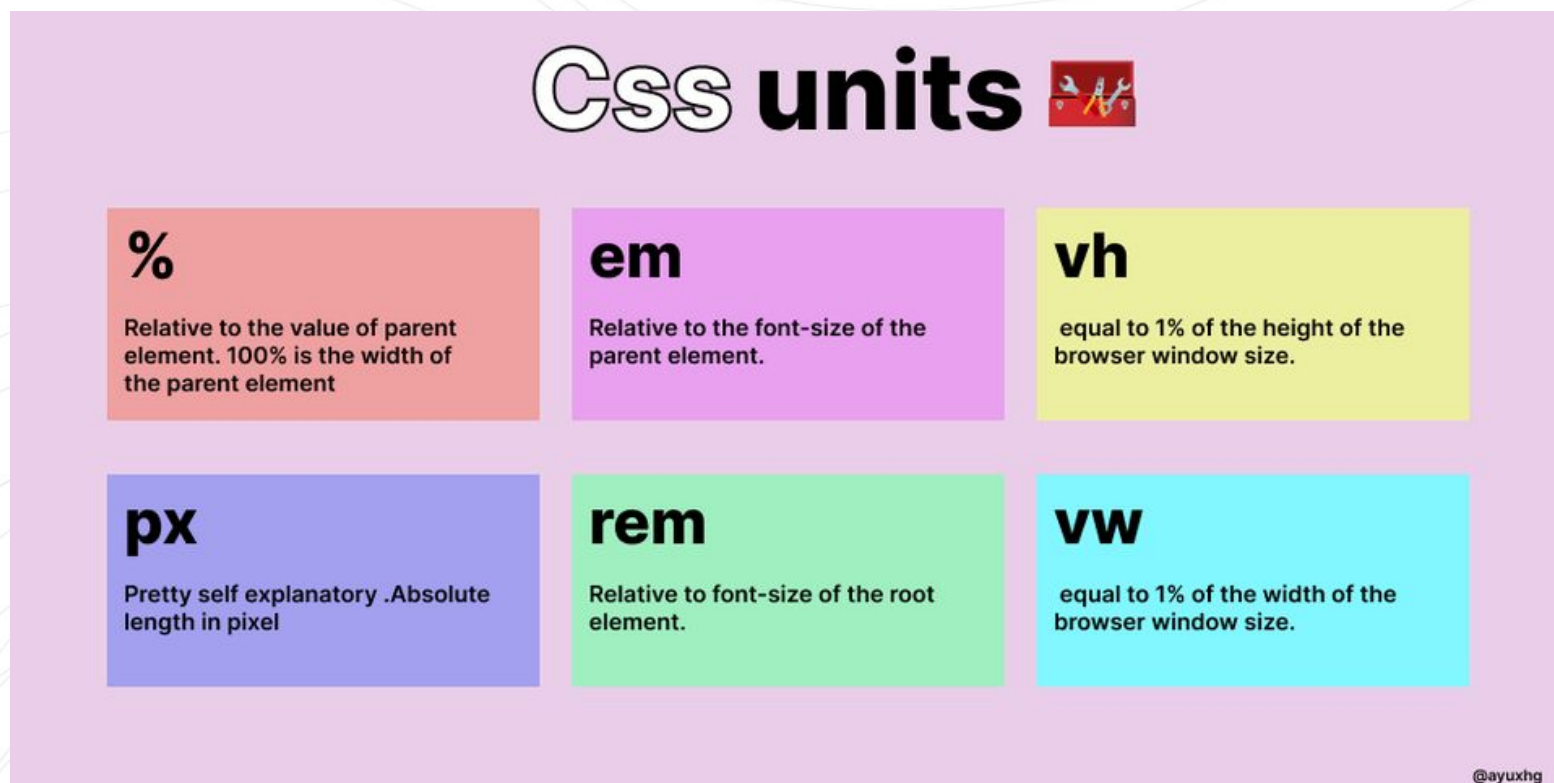
# Ćwiczenie nr 1

1. Uruchom plik **index.html** z folderu **exercises** w załączonych materiałach.
2. Wewnątrz znacznika **style** odśzukaj selektor klasy **pixels**.
3. Dla tej klasy przypisz szerokość 400px i wysokość 100px.
4. Za pomocą narzędzi developerskich zmieniaj szerokość i wysokość okna. Obserwuj jak zachowuje się niebieski box na stronie.



# Jednostki względne

Jak sama nazwa wskazuje, jednostki względne pozwalają określać wymiary elementów strony w zależności od czegoś innego (np. wielkości okna, wielkości czcionki, wielkości elementu - rodzica).



[https://www.reddit.com/r/webdev/comments/nlcejz/css\\_units\\_quicksheet/](https://www.reddit.com/r/webdev/comments/nlcejz/css_units_quicksheet/)





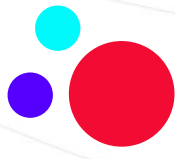
## Ćwiczenie 2 – vw / vh / vmin / vmax

1. Uruchom plik **index.html** z folderu **exercises** w załączonych materiałach.
2. Wewnątrz znacznika **style** dodaj selektor klasy **viewport**.
3. Dla tej klasy przypisuj różne – wybrane wartości width i height.
4. Postaraj się wykorzystać wszystkie dostępne jednostki (vw, vh, vmin, vmax).
5. Za pomocą narzędzi developerskich zmieniaj szerokość i wysokość okna. Obserwuj jak zachowuje się zielony box na stronie dla każdej z jednostek.



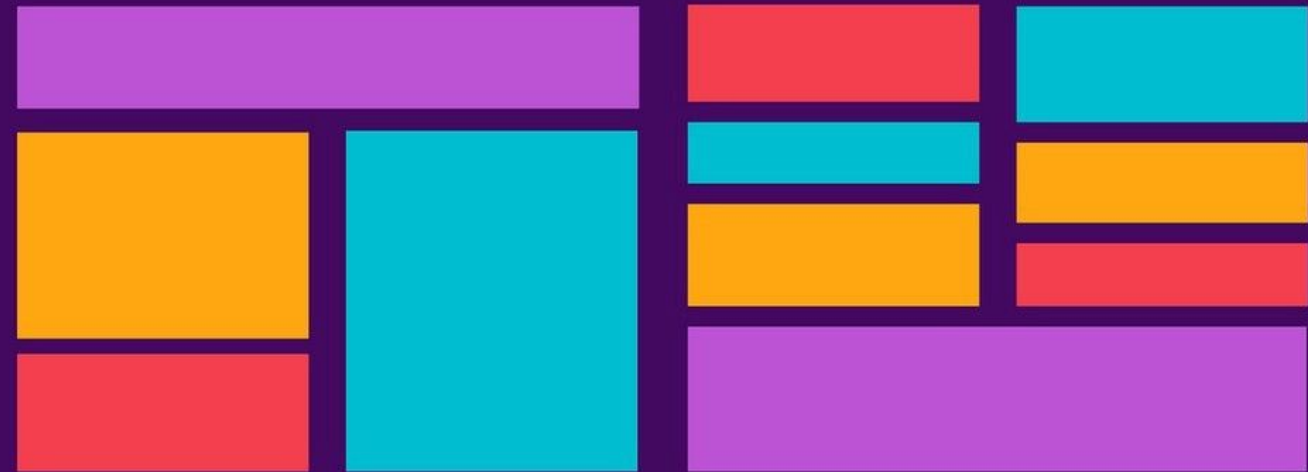
## Ćwiczenie nr 3 – em i rem

1. Uruchom plik **emrem.html** z folderu **exercises** w załączonych materiałach.
2. Wewnątrz znacznika **style** dodaj selektor klasy **em** oraz **.em > p**.
3. Ustaw wielkość czcionki **font-size: 1.5em** dla obu selektorów.
4. Wewnątrz znacznika **style** dodaj selektor klasy **rem** oraz **.rem > p**.
5. Ustaw wielkość czcionki **font-size: 1.5rem** dla obu selektorów.
6. Zmieniaj wielkość czcionki w selektorze html i porównaj zachowanie obu akapitów tekstu.
7. \* Zmień wielkość czcionki w przeglądarce i sprawdź zachowanie strony



# Ćwiczenie nr 4 – percentage (%)

1. Uruchom plik **percentage.html** z folderu **exercises** w załączonych materiałach.
2. Wewnątrz znacznika **style** odszukaj selektor klasy **box1**
3. Ustaw szerokość i wysokość elementu za pomocą jednostki (%).
4. Za pomocą narzędzi developerskich zmieniaj szerokość i wysokość okna. Obserwuj jak zachowuje się żółty box na stronie.
5. Odszukaj selektor elementu **body**. Usuń lub zakomentuj linię *height: 100%*.
6. Za pomocą narzędzi developerskich zmieniaj szerokość i wysokość okna. Obserwuj jak zmieniło się zachowanie żółtego boxa na stronie.



# CSS FLEXBOX

<https://harsh-patel.medium.com/a-complete-guide-to-flexbox-c1d2bc33d10a>





# Flexbox container



<https://sharkcoder.com/layout/flexbox>

## **display: flex (inline-flex)**

Definiuje Flex container. Po zastosowaniu na elemencie, wszystkie jego bezpośrednie dzieci będą renderowane za pomocą flexboxa. W przypadku użycia inline-flex efekt jest taki sam, z tym zastrzeżeniem, że sam kontener renderowany jest inline'owo (podobnie jak w przypadku inline-block).

## **flex-direction: row / column**

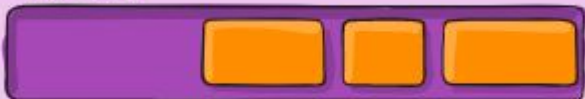
Definiuje ustawienie elementów wewnątrz flex containera. Domyślną wartością jest **row**. Co oznacza, że elementy są renderowane obok siebie. W przypadku użycia wartości **column**, elementy renderowane są jeden pod drugim.

## justify-content

flex-start



flex-end



center



space-between



space-around



space-evenly



## justify-content

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/#aa-properties-for-the-parentflex-container>

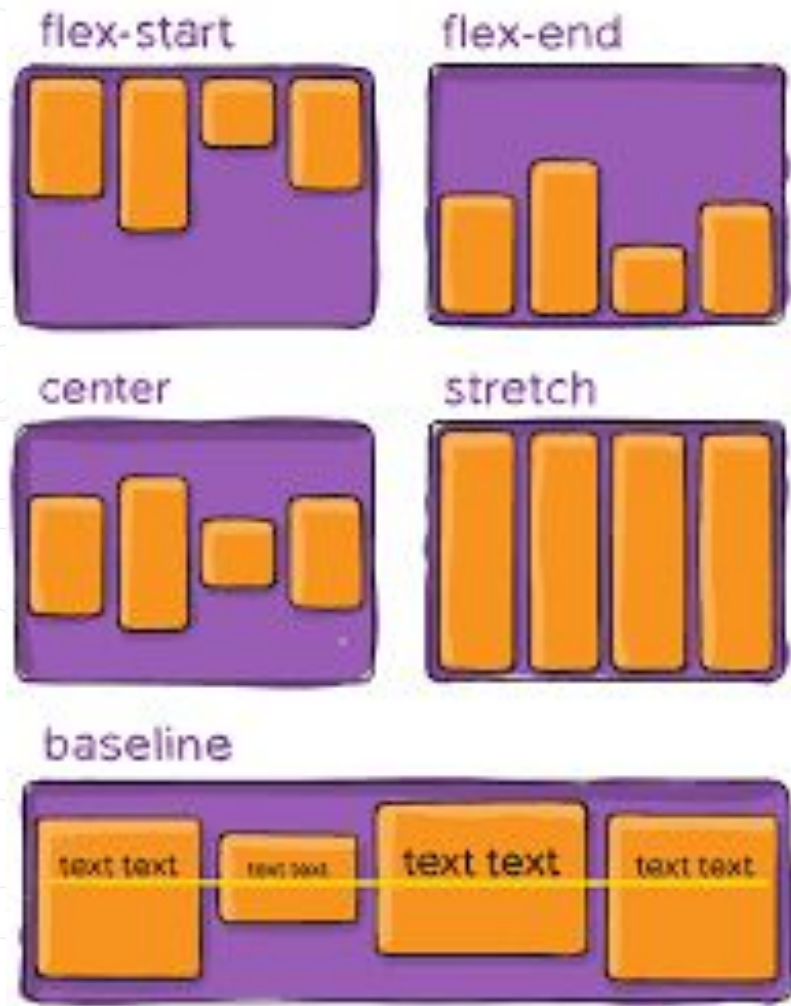




# Ćwiczenie nr 5 – Flexbox container 1

1. Uruchom plik **flexbox1.html** z folderu **exercises** w załączonych materiałach.
2. Przyjrzyj się strukturze pliku.
3. Wyrenderuj wszystkie elementy z klasą **box** w jednym rzędzie, obok siebie, w taki sposób, by wolna przestrzeń została rozdzielona po równo pomiędzy elementami, bez wolnych przestrzeni po bokach.
4. Następnie wykonaj to samo ćwiczenie, jednak ustaw elementy z klasą **box** jeden pod drugim, na dostępnej wysokości ekranu.

## align-items

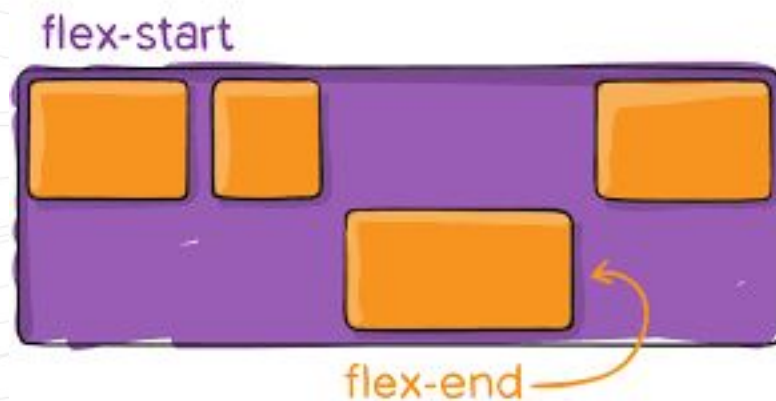


<https://css-tricks.com/snippets/css/a-guide-to-flexbox/#aa-properties-for-the-parentflex-container>



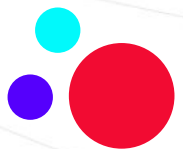
<https://sharkcoder.com/layout/flexbox>

## align-self



<https://css-tricks.com/snippets/css/a-guide-to-flexbox/#aa-properties-for-the-parentflex-container>





## Ćwiczenie nr 6 – Flexbox container 2

1. Uruchom plik **flexbox2.html** z folderu **exercises** w załączonych materiałach.
2. Przyjrzyj się strukturze pliku.
3. Wyrenderuj wszystkie elementy z klasą **box** w jednym rzędzie.
4. Wyśrodkuj wszystkie elementy w pionie i w poziomie.
5. Pierwszy element **box** w pionie powinien być wypozycjonowany do górnej krawędzi kontenera.
6. Ostatni element **box** w pionie powinien być wypozycjonowany do dolnej krawędzi kontenera.

## **flex-grow: <number>**

Przyjmuje wartość od 0 wzwyż. Parametr ten pozwala zdefiniować w jaki sposób flex item może się powiększać jeśli to możliwe (tzn jeśli istnieje wolna przestrzeń w flex containerze). Wartość służy jako proporcja względem innych itemów. Oznacza to, że jeśli wewnątrz kontenera mamy dwa itemy, z który oba mają ustawione flex-grow: 1, to oba rozszerzą się do tej samej szerokości jeśli to możliwe. Jeśli jeden z nich ustawi flex-grow: 2, to jeden element będzie się powiększać dwa razy szybciej od drugiego itd.

## **flex-shrink: <number>**

Działa na dokładnie takiej samej zasadzie jak **flex-grow**. Jak nazwa wskazuje, służy jednak do ustawiania w jaki sposób elementy powinny się zmniejszać, jeśli nie są w stanie pomieścić się wewnątrz kontenera.

## **flex-basis: <value> | auto**

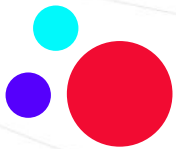
Określa wielkość elementu **zanim** wolna przestrzeń zostanie rozdystribuowana. Domyślnie ustawiona jest wartość **auto**, co oznacza, że **flex-basis** ustawi się na szerokość / wysokość elementu. Popularną praktyką jest ustawianie tej wartości na **0**, jednocześnie ustawiając **flex-grow** i **flex-shrink** na wartość  $> 0$ .





# Ćwiczenie nr 7 – Flexbox items

1. Uruchom plik **flexbox3.html** z folderu **exercises** w załączonych materiałach.
2. Przyjrzyj się strukturze pliku.
3. Wyrenderuj wszystkie elementy z klasą **box** w jednym rzędzie.
4. Elementy powinny rozciągać się na całą szerokość strony.
5. Pierwszy box powinien być 3 razy większy od ostatniego.
6. Drugi box powinien być 2 razy większy od ostatniego.



# Q & A

# Koniec

infoShare Academy