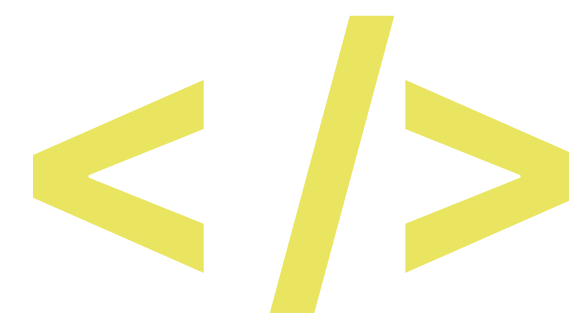


**za<pro/>gramowani.com**

**</> webmaster**



# kontakt

**E-mail:** [hello@filiposinski.com](mailto:hello@filiposinski.com)

**Discord:** <https://discord.gg/eyhAtuxJcv>

**GitHub:** [github.com/zaprogramowaniFO](https://github.com/zaprogramowaniFO)

# Zajęcia **nr 4**

2023/11/14

**</> DIV vs SPAN vs SECTION**

## Różnice

### <div>

- pojemnik na treść;
- nie ma znaczenia semantycznego;
- używany do grupowania zawartości;
- element blokowy;

### <span>

- pojemnik na treść;
- nie ma znaczenia semantycznego;
- używany do stylizacji tekstu;
- element liniowy;

### <section>

- pojemnik na treść;
- ma znaczenie semantyczne;
- służy do tematycznego opakowywania;

# **CSS BOX MODEL**



## CSS BOX MODEL

Model pudełkowy pozwala przedstawiać i opisywać treść. Pozwala na określanie wyglądu i układu poszczególnych elementów.

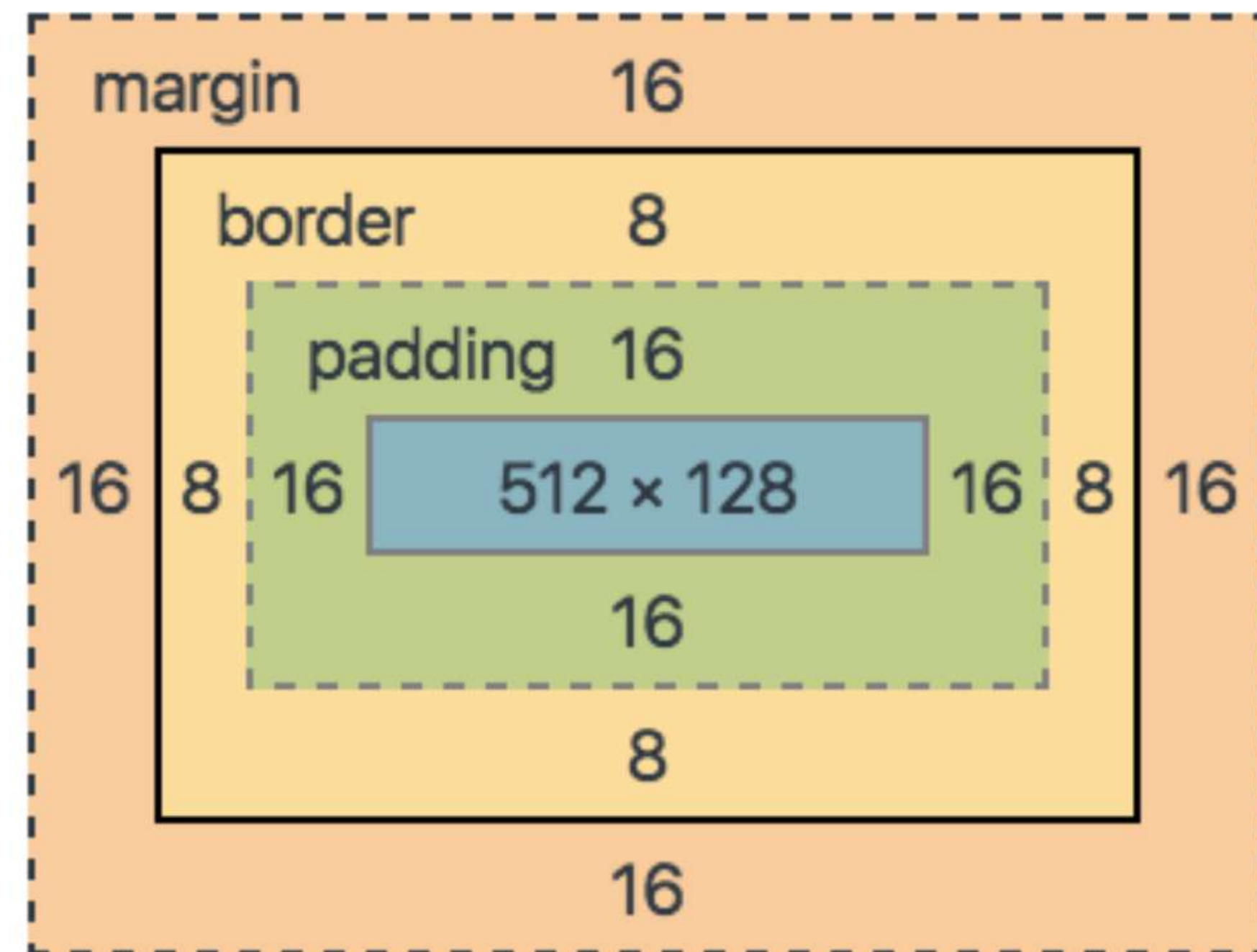
Na model pudełkowy składają się poniższe elementy:

**Niebieski kolor to nasza treść.**

**Kolor zielony to wewnętrzny margines.**

**Żółty to grubość ramki.**

**Czerwony to zewnętrzny margines.**





## Argumenty w selektorach



**padding: top right bottom left;** wymiary wew. marginesu;  
**margin: top right bottom left;** wymiary zew. marginesu;

Każdy z wymiarów możemy opisać pojedynczo, zgrupować w pary lub podać jednej właściwości wszystkie.

Należy pamiętać o zasadzie zegara.

## Border, czyli ramka

**Border: \*rozmiar\* ;**

grubość ramki

**Border-style: \*wybrany styl\*;**

zdefiniowanie stylu ramki, np. Solid, dotted, dashed

**border-color: \*kolor\*;**

zdefiniowanie koloru ramki

**Border-radius: \*wartość zaokrąglenia narożników\*;**

zaokrąglenie narożników

**Border: \*wymiar\* \*styl\* \*kolor\*;**

można również zdefiniować ramkę w ten sposób

Możemy również zdefiniować ramkę na wiele sposobów.

**</> JEDNOSTKI**

## Jednostki

W zależności od naszych potrzeb rozróżniamy różne jednostki miary, które możemy użyć w naszych projektach, m.in.:

- jednostki relatywne względem fontów
- jednostki relatywne względem okna (viewport)
- jednostki relatywne względem rodzica
- jednostki absolutne
- wartości bezjednostkowe

## Jednostki relatywne względem fontów

**em** - odwołuje się do bezpośredniego rodzica elementu

**rem** - odwołuje się do globalnej wielkości tekstu

## Jednostki relatywne względem fontów

**vh** - viewport height, relatywny względem wysokości okna

**vw** - viewport width, relatywny względem szerokości okna

**vmin** - minimum relatywne względem okna

**vmax** - maksimum relatywne względem okna

## **Jednostka relatywna względem rodzica**

%- jednostka procentowa relatywna względem rodzica

## **Jednostka absolutna, niezależna względem innych elementów**

px - jednostka absolutna niezależna od innych elementów

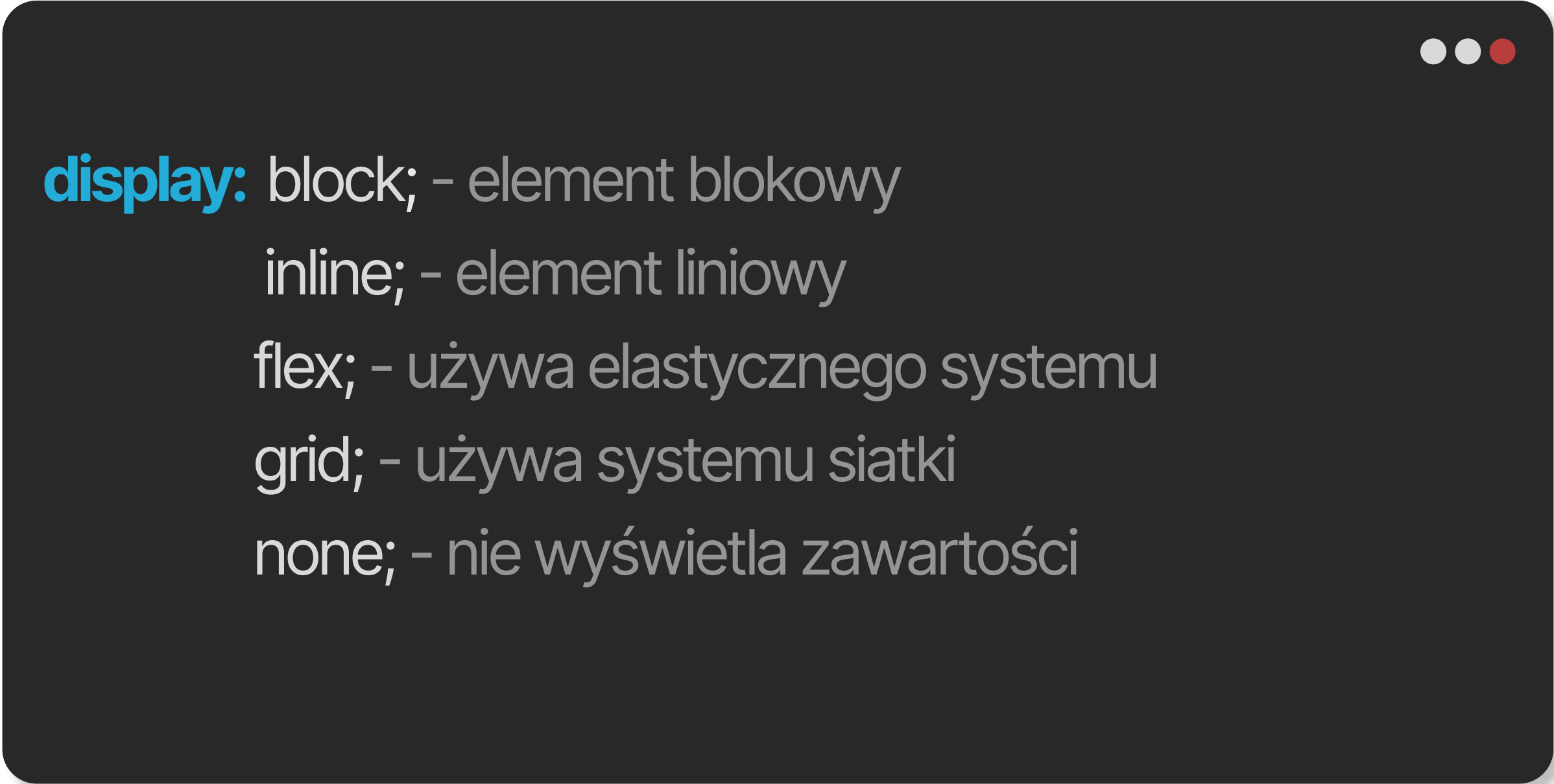
## Wartości bezjednostkowe

**margin: 0;**  
**opacity: 1;**  
**line-height: 1.5;**



**</> DISPLAY**

# WŁAŚCIWOŚCI



**display:** block; - element blokowy  
inline; - element liniowy  
flex; - używa elastycznego systemu  
grid; - używa systemu siatki  
none; - nie wyświetla zawartości

Właściwość display odpowiada za sposób wyświetlania elementu. Określenie czy dany element ma być traktowany jako element **blokowy** czy **liniowy**. Możemy również w ten sposób ukryć dany element.

Do najważniejszych należą wymienione obok.

Należy pamiętać, że wartości takie jak **flex** lub **grid** mają dodatkowe właściwości.

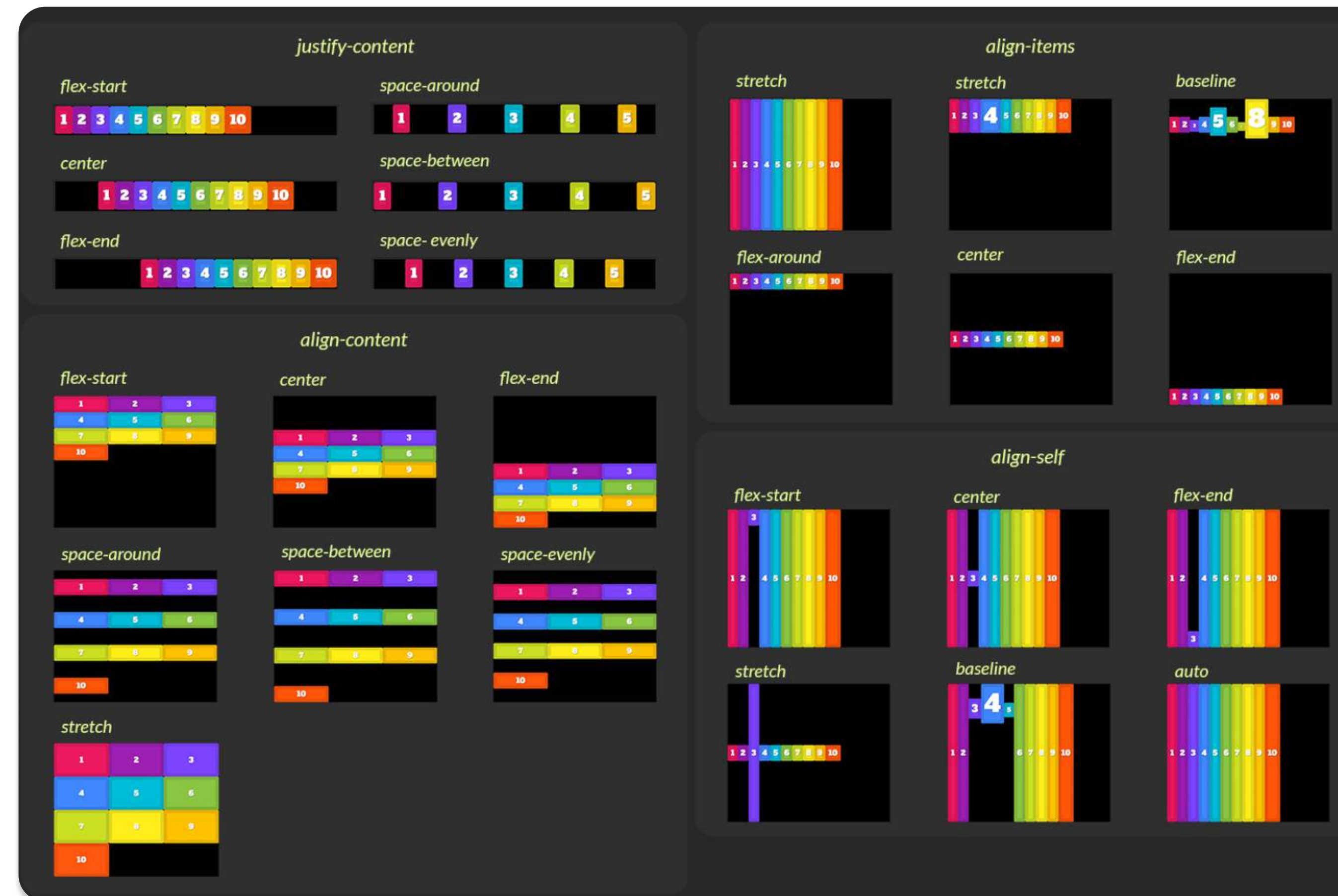
**</> FLEXBOX**

# FLEXBOX

Flexbox pozwala na tworzenie elastycznych, responsywnych szablonów stron.

Aby dany element korzystał z szablonu flexbox, musimy dodać do jego stylu poniższą właściwość:

**display: flex;**



## WŁAŚCIWOŚCI

**flex-direction:** \*kierunek zawartości\* ;

**flex-wrap:** \*zawijanie zawartości\* ;

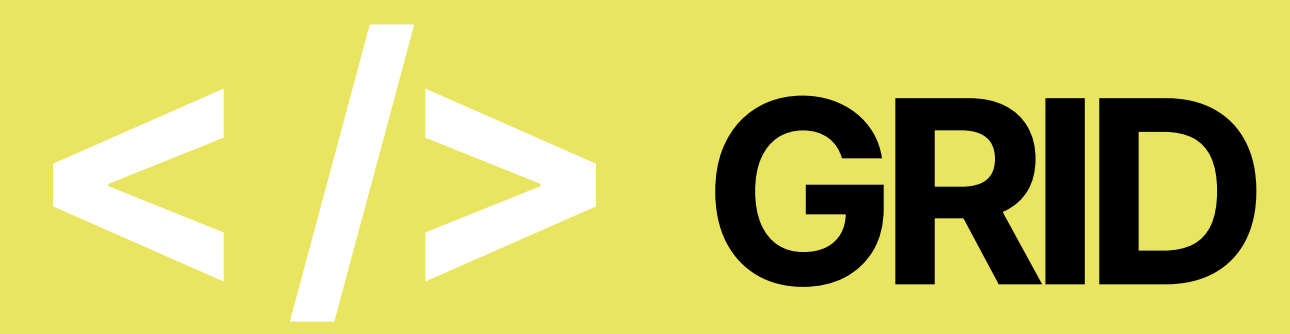
**justify-content:** \*środkowanie zawartości\* ;

**align-content:** \*środkowanie zawartości\* ;

**align-items:** \*środkowanie zawartości\* ;

Właściwości pozwalające ustalić kierunek wyświetlania elementów oraz ich zawijanie.

Właściwości pozwalające ustalić pozycjonowanie elementów względem rodzica.

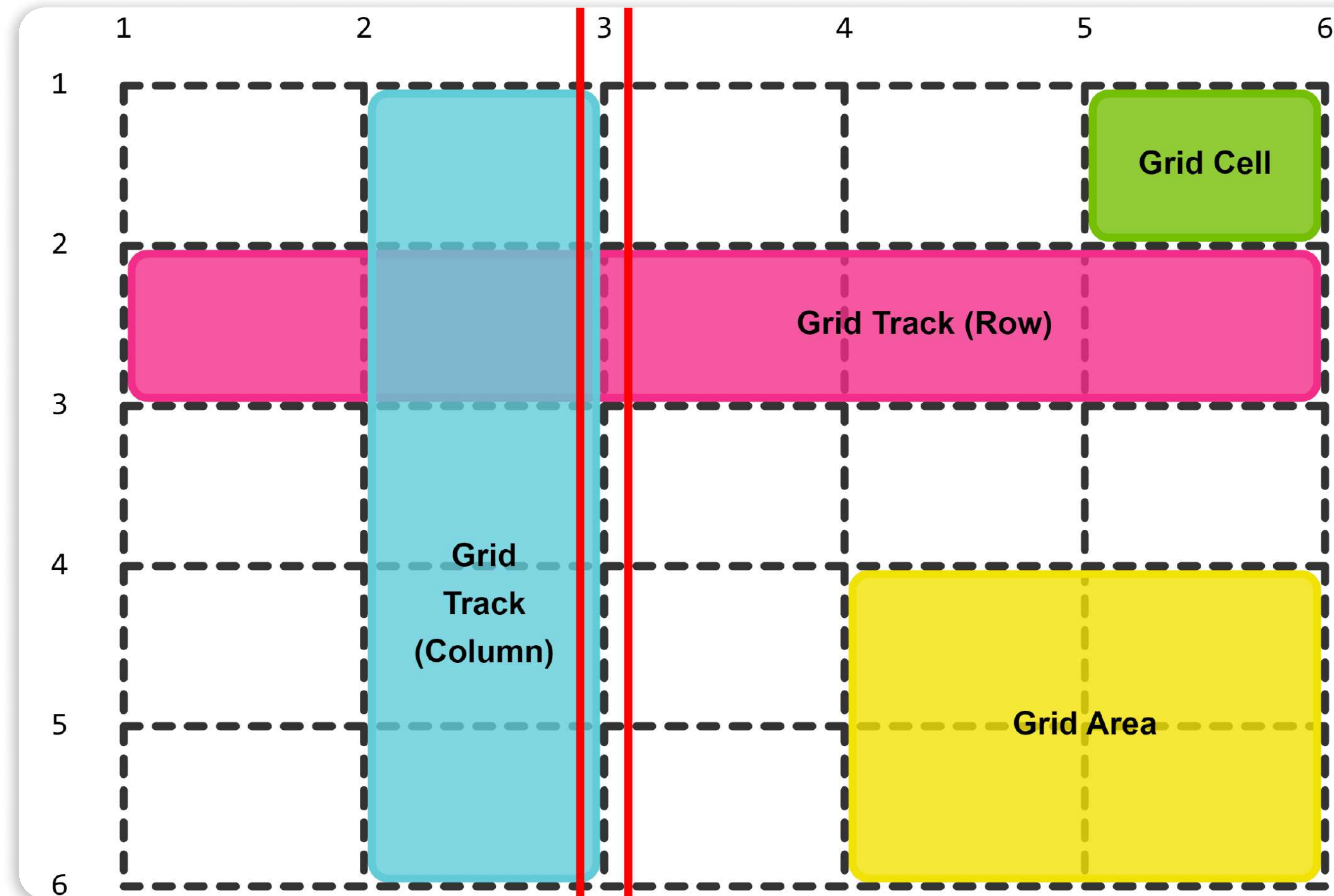


## GRID

Grid layout służy do budowania szablonu strony opartego na siatce. Jest odpowiedzią na potrzebę stworzenia prostego systemu budowania layoutów stron.

Aby wykorzystać szablon należy dodać do właściwości elementu poniższą właściwość:

**display: grid;**





## WŁAŚCIWOŚCI

```
grid-template-columns: *podział na kolumny*;  
grid-template-rows: *podział na wiersze*;  
grid-template: *podział wierszy* / *podział kolumny*;
```

Właściwości pozwalające ustalić ilość kolumn oraz wierszy w naszej siatce:

## WŁAŚCIWOŚCI

```
grid-column-gap: *wartość*;
```

```
grid-row-gap: *wartość*;
```

```
grid-gap: *wartość wierszy* *wartość kolumn*;
```

Własności pozwalające ustalić wielkość przerwy pomiędzy elementami siatki.

## WŁAŚCIWOŚCI

```
grid-column-start: *kolumna startowa*;
grid-column-end: *kolumna końcowa*;

grid-column: *kolumna startowa* / *kolumna końcowa*;
```

Właściwości pozwalające ustalić początkową i końcową kolumnę elementu. Ważne: element skończy się na kolumnie wcześniejszej.

## WŁAŚCIWOŚCI

```
grid-row-start: *wiersz startowy*;
grid-row-end: *wiersz końcowy*;

grid-row: *wiersz startowy* / *wiersz końcowy*;
```

Właściwości pozwalające ustalić początkowy i końcowy wiersz elementu. Ważne: element skończy się na wierszu wcześniej.

## WŁAŚCIWOŚCI

```
grid-area: *startowy wiersz* / * startowa kolumna* /  
*końcowy wiersz* / *końcowa kolumna*;
```

Właściwość pozwalająca stworzyć całą płaszczyznę, na podobnych zasadach jak wcześniej.

## WŁAŚCIWOŚCI

```
grid-template-areas: "a a a"  
                    "b c c"  
                    "b c c";  
  
grid-area: a;
```

Właściwość, która pozwala na stworzenie szablonu na naszej siatce.  
Następnie we właściwościach każdego z elementów musimy dodać odwołanie do danej komórki.

**</> GRY DO NAUKI**



## GRY WSPOMAGAJĄCE NAUKĘ

### GRID GARDEN

<https://cssgridgarden.com>

### FLEXBOX FROGGY

<https://flexboxfroggy.com>

### FLEXBOX DEFENCE

<http://www.flexboxdefense.com>

## GRID GARDEN

Poziom 1 z 28


Witaj w grze Grid Garden, w której pisząc kod CSS rozwijasz swój ogród z marchewkami! Podlewaj tylko pola z marchewkami używając właściwości **grid-column-start**.

Na przykład, **grid-column-start: 3;** nawodni obszar zaczynający się od trzeciej pionowej linii, co jest innym sposobem na powiedzenie trzecia od lewej pionowa granica.

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20% 20%;
4   grid-template-rows: 20% 20% 20% 20% 20%;
5 }
6
7 #water {
8   |
9 }
10
11
12
13
14
```

Dalej

Grid Garden jest wytworem [Codepip](#) • [GitHub](#) • [Twitter](#) • [Polski](#)



## FLEXBOX FROGGY

Poziom 1 z 24

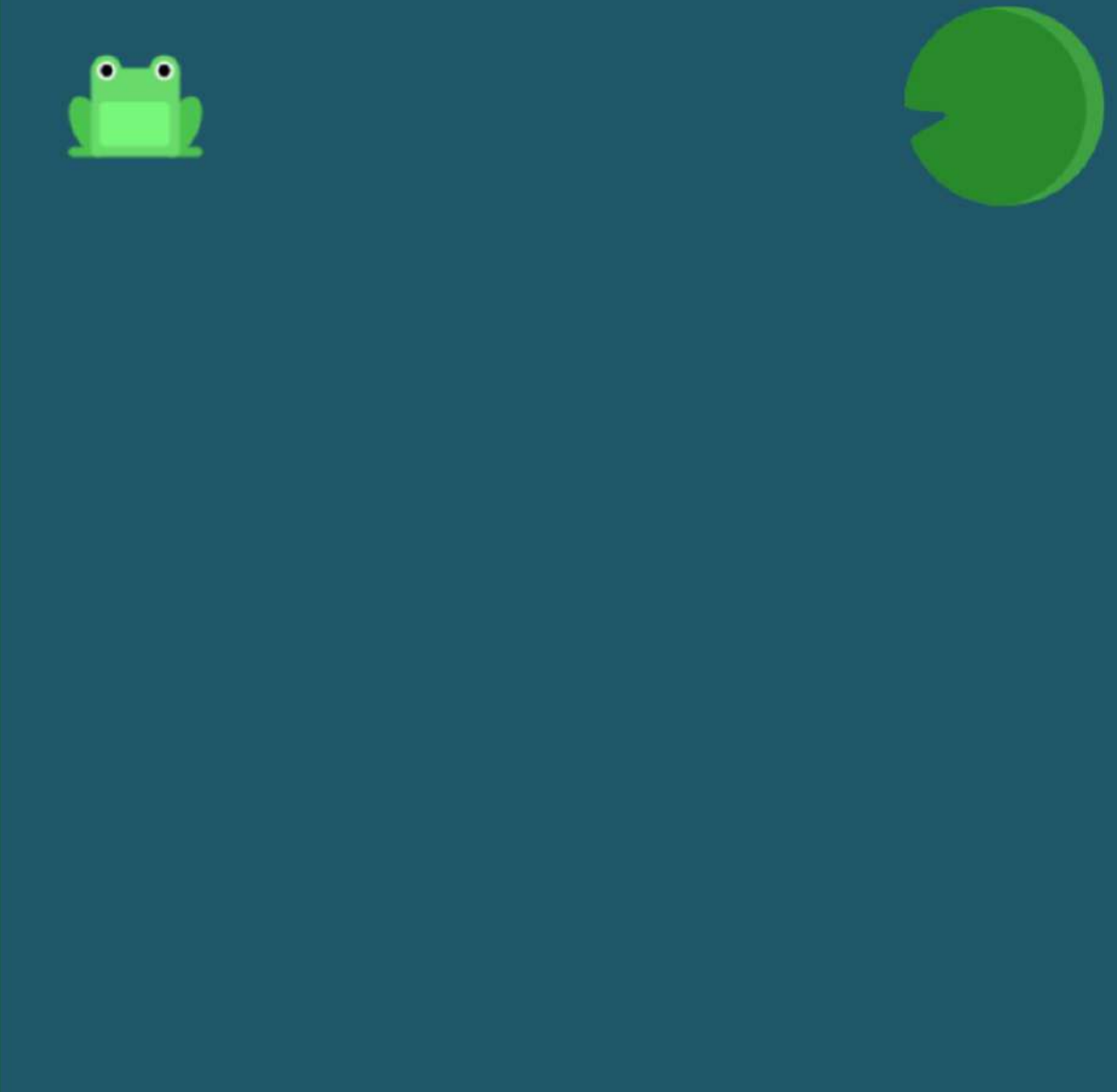
Witaj w grze Flexbox Froggy. Grze, w której pomagasz żabce Froggy i jej przyjaciółom pisać kod CSS! Doprowadź żabki do odpowiednich listków przy pomocy **justify-content**, właściwości która wyrównuje elementy w poziomie i przyjmuje wartości:

- flex-start**: Elementy wyrównują się do lewej strony kontenera.
- flex-end**: Elementy wyrównują się do prawej strony kontenera.
- center**: Elementy wyrównują się do środka kontenera.
- space-between**: Elementy wyświetlają się na całej szerokości kontenera z równymi odstępami.
- space-around**: Każdy z elementów wyświetla się z taką samą ilością przestrzeni wokół.

Na przykład: **justify-content: flex-end;** przesunie żabę do prawej strony.

```
1 #pond {
2   display: flex;
3   |
4 }
5
6
7
8
9
10
```

Dalej



**</> RESET i NORMALIZACJA**

## Reset i normalizacja

<https://necolas.github.io/normalize.css/>

Każda przeglądarka ma własne domyślne style CSS. Powoduje to, że nasza strona będzie na każdej przeglądarce wyglądać trochę inaczej.

Dlatego dobrą praktyką jest resetowanie ustawień domyślnych aby uspójnić wygląd.

Aktualnie, najpopularniejszym rozwiązaniem, jest użycie **normalizacji**, która standaryzuje interpretację znaczników przez różne przeglądarki.



**Normalize.css**

## Reset i normalizacja

```
html {  
    box-sizing: border-box;  
}  
  
*,  
*:after,  
*:before {  
    box-sizing: inherit;  
}
```

Dodatkowo, możemy dołożyć w naszym pliku głównym CSS właściwość, która będzie dziedziczyć pozwalając na dziedziczenie wielkości przez elementy.