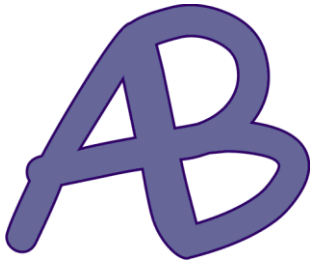
Wykład



GRID

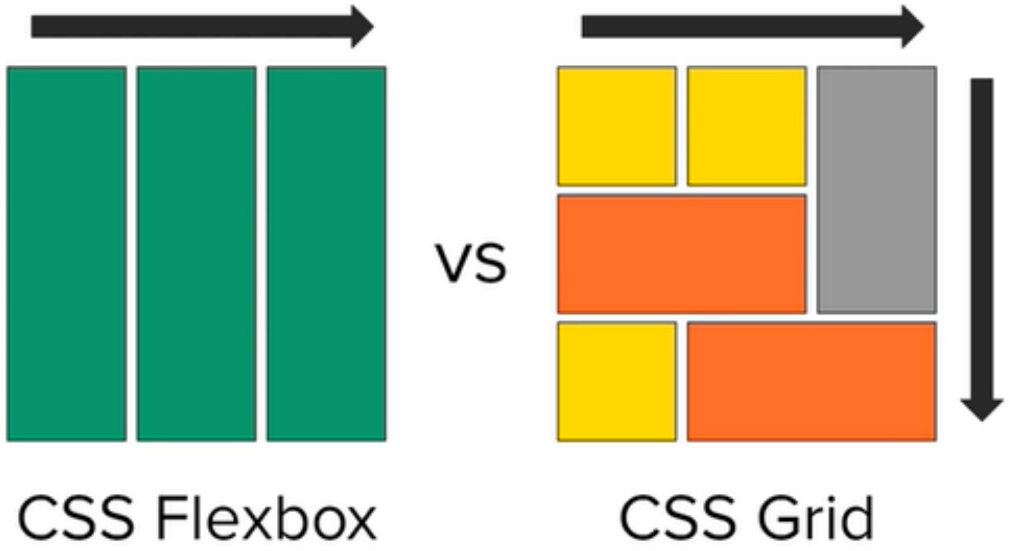
CSS Grid Layout

CSS Grid Layout to układu oparty na siatce, z wierszami i kolumnami, co ułatwia projektowanie stron internetowych..

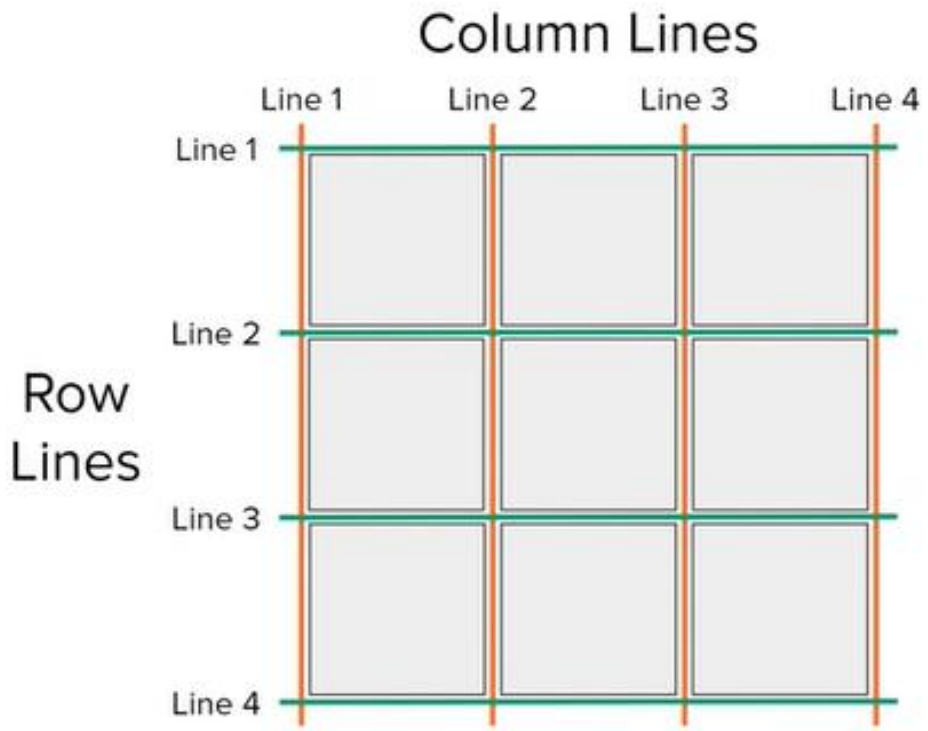
CSS Grid Layout jest najlepszą metodą aby podzielić stronę na części, definiowanie relacji takich jak rozmiar, pozycja i warstwa, pomiędzy podstawowymi elementami HTML.

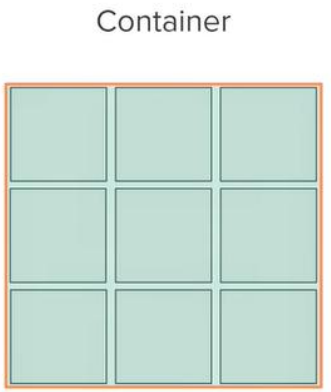
Podobnie do tabela rozmieszczenie grid pozwala układać elementy w kolumny i wiersze. Jednak grid dopuszcza więcej ułożeń niż tabela. Na przykład kontener grid potrafi spozycjonować elementy wewnątrz siebie w taki sposób, aby na siebie nachodziły oraz używa warstw.

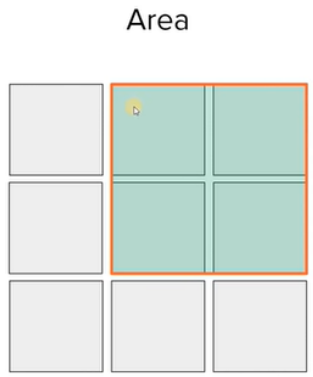
CSS Grid Layout

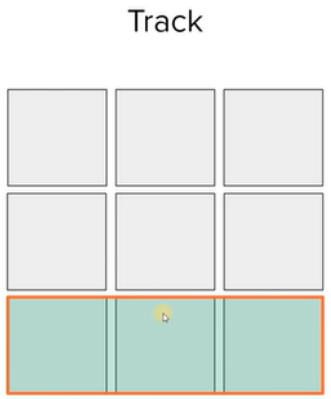


Podstawowe pojęcia







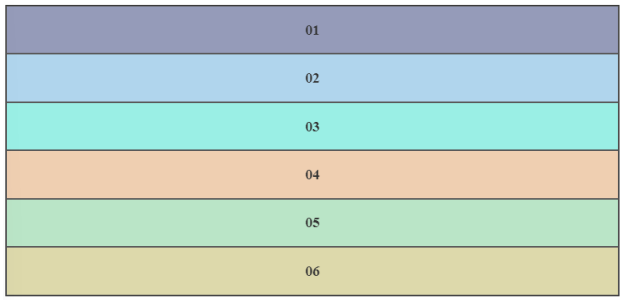


CSS Grid Layout

Kod na którym będą bazować poniższe przykłady

```html
<section class="grid_container">
   <article class="item_1">
     <h3>01</h3>
   </article>
   <article class="item_2">
     <h3>02</h3>
   </article>
   <article class="item_3">
     <h3>03</h3>
   </article>
   <article class="item_4">
     <h3>04</h3>
   </article>
   <article class="item_5">
     <h3>05</h3>
   </article>
   <article class="item_6">
     <h3>06</h3>
   </article>
 </section>
```

```css
section {
margin: 10px;
border: 1px solid rgb(47, 47, 47);
}
article {
  display: flex;
  align-items: center;
  justify-content: center ;
  border: 1px solid rgb(47, 47, 47);
}
article {
  filter: opacity(80%);
}
```




```css
.grid_container {
}
.item_1 {
  background-color: #7a82a7;
}
.item_2  {
  background-color: #9ccae9;
}
.item_3  {
  background-color: #81ebde;
}
.item_4  {
  background-color: #ebc39d;
}
.item_5  {
  background-color: #a9dfb9;
}
.item_6  {
  background-color: #d5cf96;
}
```

Definiowanie wierszy i kolumn

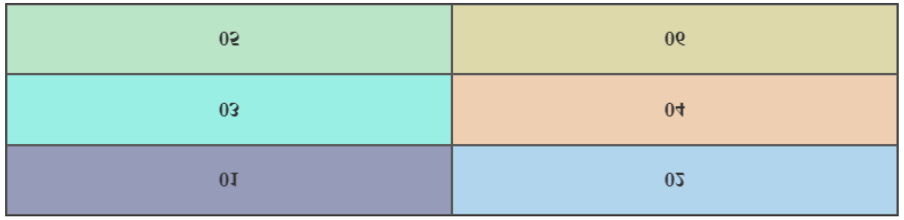
```
.grid_container {
  display: grid;
  grid-template-columns: 1fr 1fr;
}
```

01	02
03	04
05	06

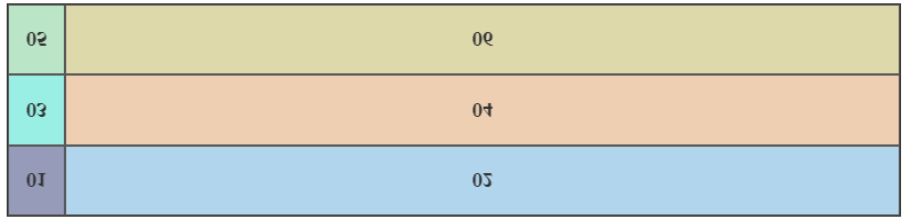
CSS Grid Layout

Definiowanie wierszy i kolumn

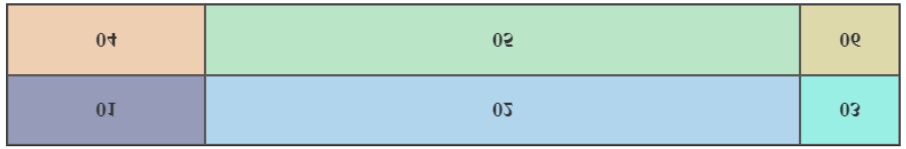
```
.grid_container {
  display: grid;
  grid-template-columns: 1fr 1fr;
}
```



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



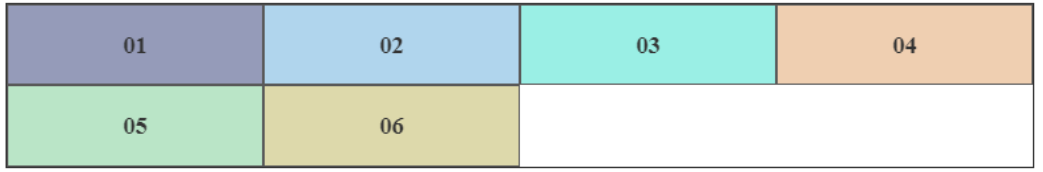
```
.grid_container {
  display: grid;
  grid-template-columns:
1fr 3fr .5fr;
}
```



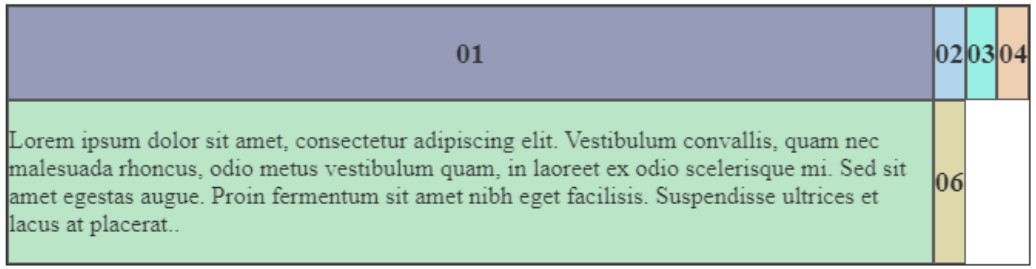
CSS Grid Layout

Definiowanie wierszy i kolumn

```
.grid_container {
  display: grid;
  grid-template-columns: repeat(4,1fr);
}
```



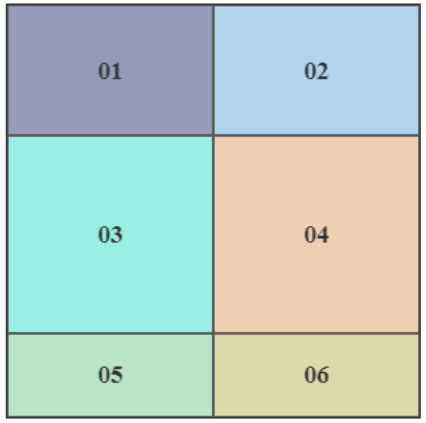

```
.grid_container {
  display: grid;
  grid-template-columns: repeat(4,
auto);
}
```



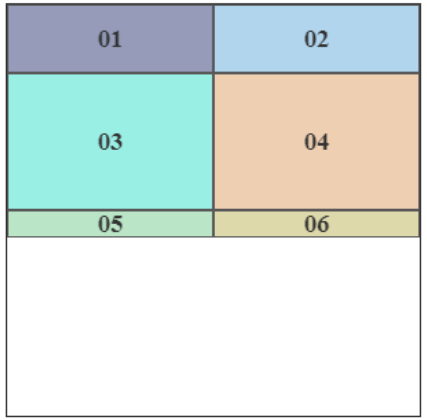
CSS Grid Layout

Definiowanie wierszy i kolumn

```
.grid_container {
  display: grid;
  height: 300px;
  width: 300px;
  grid-template-columns: repeat(2,1fr);
  grid-template-rows: 2fr 3fr 1fr;
}
```

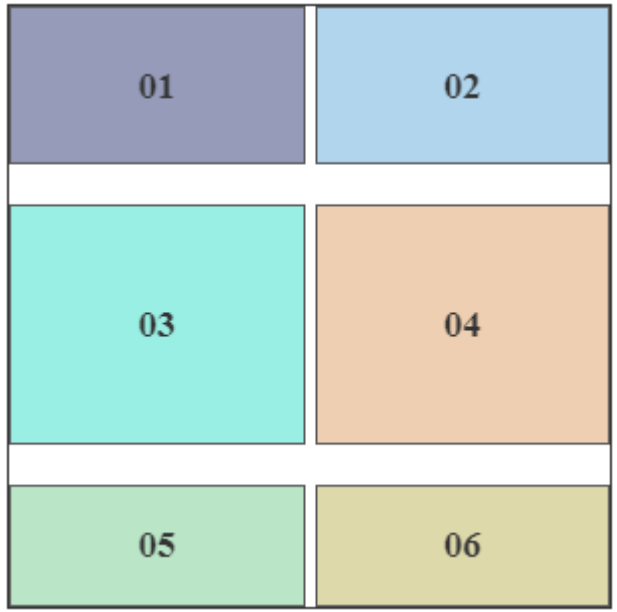



```
.grid_container {
  display: grid;
  height: 300px;
  width: 300px;
  grid-template-columns: repeat(2,1fr);
  grid-template-rows: 50px 100px 20px;
}
```



Odstępy pomiędzy kolumnami i wierszami

```
gap: 20px 5px;

  column-gap: 5px;
  row-gap: 20px;
```

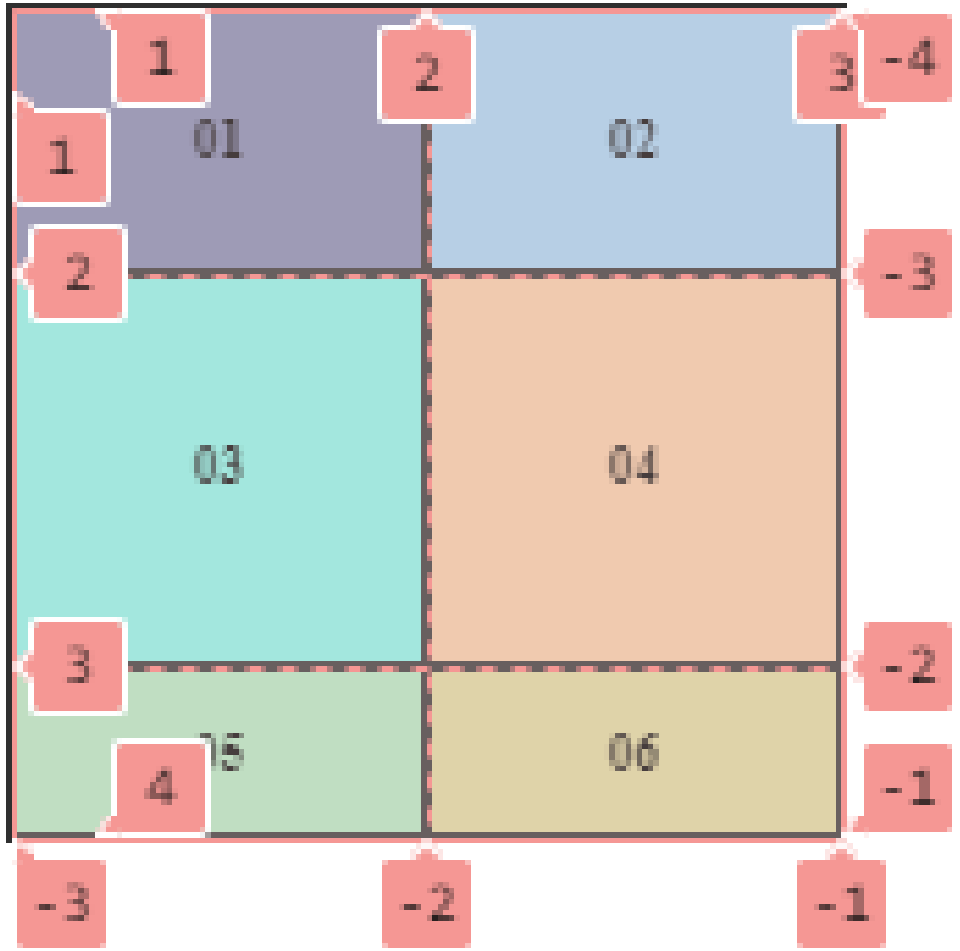


Zwróćmy uwagę, że numeracja linii kolumn i linii wierszy jest dwustronna:

- - od lewej do prawej liczby dodatnie (od jedynki)
- od prawej do lewej liczby ujemne (od -1)

Gdzie -1 oznacza ostatnią linię -2 przedostatnią i tak dalej.

Uwaga! Rozmieszczając elementy odnosimy się nie do kolumn i wierszy lecz bezpośrednio do linii.



Rozmieszczanie elementów sposób pierwszy: względem linii kolumn i linii wierszy

```
<section class="grid_container">
  <article class="item_1">
    <h3>01</h3>
  </article>
  <article class="item_2">
    <h3>02</h3>
  </article>
  <article class="item_3">
    <h3>03</h3>
  </article>
  <article class="item_4">
    <h3>04</h3>
  </article>
  <article class="item_5">
    <h3>05</h3>
  </article>
  <article class="item_6">
    <h3>06</h3>
  </article>
</section>
```

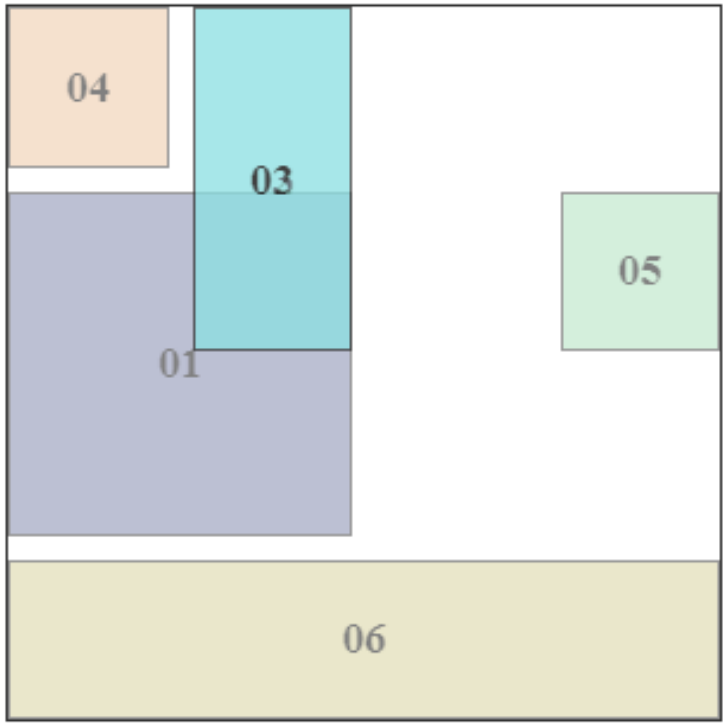
```
section {
margin: 10px;
border: 1px solid rgb(47, 47, 47);
}

article {
  display: flex;
  align-items: center;
  justify-content: center ;
  border: 1px solid rgb(47, 47, 47);
}
article {
  filter: opacity(50%);
}
```

```
.grid_container {
  display: grid;
  height: 300px;
  width: 300px;
  grid-template-columns: repeat(4,1fr);
  grid-template-rows: repeat( 4, 1fr);
  gap: 10px;
}
```

Rozmieszczanie elementów sposób pierwszy: względem linii kolumn i linii wierszy

```
.item_1 {
  background-color: #7a82a7;
  grid-column-start: 1;
  grid-column-end: 3;
  grid-row-start: 2;
  grid-row-end: 4;
}

.item_2  {
  background-color: #9ccae9;
  grid-column-start: 2;
  grid-column-end: 3;
  grid-row-start: 1;
  grid-row-end: 3;
}

.item_3  {
  background-color: #81ebde;
  grid-column-start: 2;
  grid-column-end: 3;
  grid-row-start: 1;
  grid-row-end: 3;
}
```

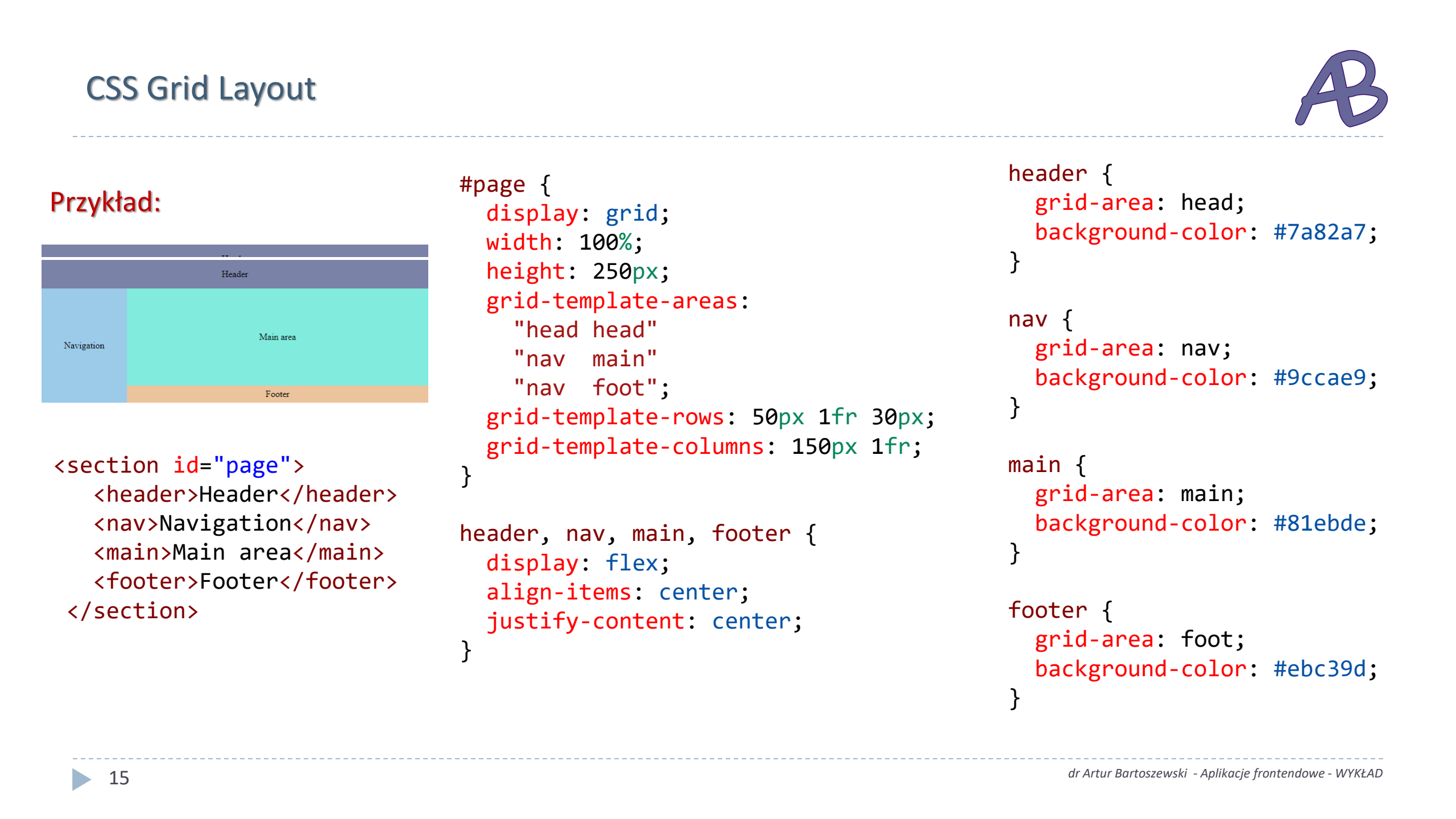
```
.item_4  {
  background-color: #ebc39d;
}

.item_5  {
  background-color: #a9dfb9;
  grid-column: -2 / -2;
  grid-row: 2 / 2;
}

.item_6  {
  background-color: #d5cf96;
  grid-column: 1 / -1;
  grid-row: -1 / -2;
}
```



grid-template-areas Właściwość ta określa nazwane obszary siatki, ustanawiając komórki w siatce.

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

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

CSS Grid Layout



Przykład:

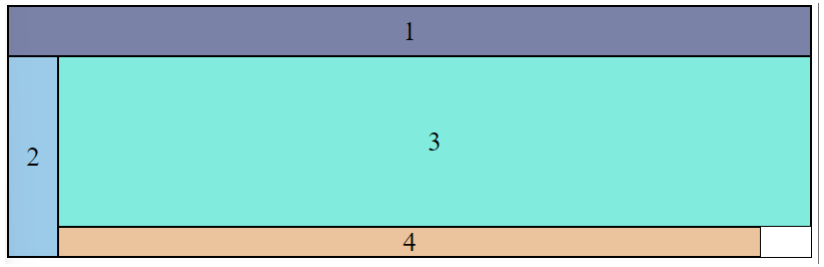

```
<section id="page">
   <header>Header</header>
   <nav>Navigation</nav>
   <main>Main area</main>
   <footer>Footer</footer>
 </section>
```

```
#page {
  display: grid;
  width: 100%;
  height: 250px;
  grid-template-areas:
    "head head"
    "nav  main"
    "nav  foot";
  grid-template-rows: 50px 1fr 30px;
  grid-template-columns: 150px 1fr;
}

header, nav, main, footer {
  display: flex;
  align-items: center;
  justify-content: center;
}
```

```
header {
  grid-area: head;
  background-color: #7a82a7;
}

nav {
  grid-area: nav;
  background-color: #9ccae9;
}

main {
  grid-area: main;
  background-color: #81ebde;
}

footer {
  grid-area: foot;
  background-color: #ebc39d;
}
```

CSS Grid Layout

II wersja:




```
<section class="contener">
   <div class="item_1">1</div>
   <div class="item_2">2</div>
   <div class="item_3">3</div>
   <div class="item_4">4</div>
 </section>
```

```
.contener {
  display: grid;
  width: 100%;
  height: 250px;
  border: 1px solid black;
  grid-template-areas:
    "head head head"
    "nav  main main"
    "nav  foot .";
  grid-template-rows: 50px 1fr 30px;
  grid-template-columns: 50px 1fr 50px;
}

div {
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 20pt;
  border: 1px solid black;
}
```

```
.item_1 {
  grid-area: head;
  background-color: #7a82a7;
}

.item_2  {
  grid-area: nav;
  background-color: #9ccae9;
}

.item_3  {
  grid-area: main;
  background-color: #81ebde;
}

.item_4  {
  grid-area: foot;
  background-color: #ebc39d;
}
```

Polecenie w rodzaju:

```
grid-template-columns: repeat(12, minmax(200px, 1fr));
```

Może spowodować przepełnienie w wierszu.
Kolumny nie zostaną zawijane w nowe wiersze, jeśli szerokość widocznego obszaru jest zbyt wąska, aby zmieścić je wszystkie z nowym wymaganiem minimalnej szerokości, ponieważ wyraźnie mówimy przeglądarce, aby powtarzała kolumny 12 razy w wierszu.

```
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
```

auto-fit
auto-fill

To słowa kluczowe informują przeglądarkę, aby obsługiwała rozmiar kolumn i zawijanie elementów za nas, dzięki czemu elementy będą zawijane w wiersze, gdy szerokość nie jest wystarczająco duża, aby zmieścić je bez przepełnienia

CSS Grid Layout

```
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
```

- minmax(min, max) określa minimalną i maksymalną szerokość kolumny.
- 1fr oznacza "jedną część" dostępnego miejsca.
- auto-fit jest preferowany, gdy chcemy, by siatka była bardziej responsywna i elastyczna.
- auto-fill bywa przydatny, gdy chcemy zachować stały układ, niezależnie od liczby elementów.

auto-fit
auto-fill

CSS Grid Layout

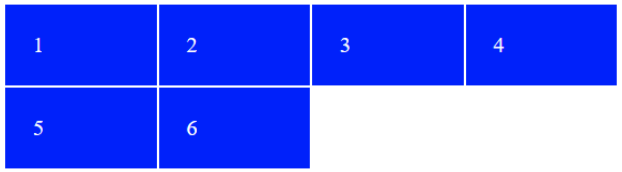
```
<h2>auto-fill</h2>
<div class="grid-container grid-container--
fill">
  <div class="grid-element">
    1
  </div>
  <div class="grid-element">
    2
  </div>
  <div class="grid-element">
    3
  </div>
  <div class="grid-element">
    4
  </div>
  <div class="grid-element">
    5
  </div>
  <div class="grid-element">
    6
  </div>
</div>
<hr>
```

```
<h2>auto-fit</h2>
<div class="grid-container grid-container--
fit">
  <div class="grid-element">
    1
  </div>
  <div class="grid-element">
    2
  </div>
  <div class="grid-element">
    3
  </div>
  <div class="grid-element">
    4
  </div>
  <div class="grid-element">
    5
  </div>
  <div class="grid-element">
    6
  </div>
</div>
```

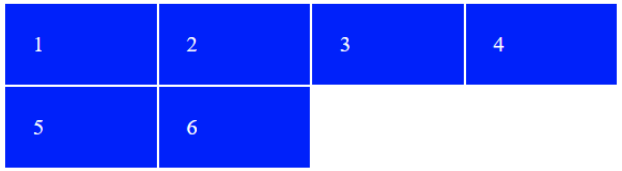
CSS Grid Layout

```
.grid-container {
    display: grid;
}

.grid-container--fill {
    grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
}

.grid-container--fit {
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
}
.grid-element {
  background-color: blue;
  padding: 20px;
  color: #fff;
  border: 1px solid #fff;
}
```

auto-fill



auto-fit



auto-fill

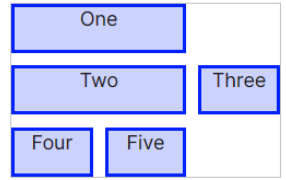
auto-fit

`grid-auto-flow`:

Właściwość CSS grid-auto-flow steruje działaniem algorytmu automatycznego umieszczania, określając dokładnie, w jaki sposób automatycznie umieszczane elementy są wlewane do siatki.

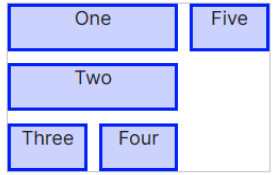
`grid-auto-flow: row;`

Elementy są umieszczane, wypełniając każdy wiersz po kolei, dodając nowe wiersze w razie potrzeby.



`grid-auto-flow: column;`

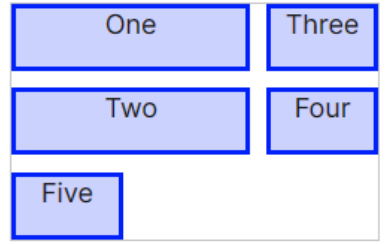
Elementy są umieszczane przez wypełnianie każdej kolumny po kolei, dodając nowe kolumny w razie potrzeby.



Przykłady ze strony: grid-auto-flow - CSS: Kaskadowe arkusze stylów | MDN (mozilla.org)

```
grid-auto-flow: row dense;
```

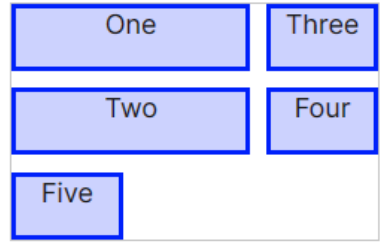
"Gęsty" algorytm pakowania próbuje wypełnić wcześniej w siatce, jeśli mniejsze elementy pojawią się później. Może to spowodować, że przedmioty będą wyglądać na nieuporządkowane, podczas gdy wypełni to pozostawione przez większe przedmioty.



Algorytm "rzadki„ (domyślny), - algorytm umieszczania przesuwa się tylko "do przodu" w siatce podczas umieszczania elementów, nigdy nie cofając się w celu wypełnienia otworów. Gwarantuje to, że wszystkie automatycznie umieszczone przedmioty będą wyświetlane "w kolejności", nawet jeśli pozostawi to, które mogłyby zostać wypełnione przez późniejsze przedmioty.

`grid-auto-flow;`

"Gęsty" algorytm pakowania próbuje wypełnić wcześniej w siatce, jeśli mniejsze elementy pojawią się później. Może to spowodować, że przedmioty będą wyglądać na nieuporządkowane, podczas gdy wypełni to pozostawione przez większe przedmioty.



Algorytm "rzadki„ (domyślny), - algorytm umieszczania przesuwa się tylko "do przodu" w siatce podczas umieszczania elementów, nigdy nie cofając się w celu wypełnienia otworów. Gwarantuje to, że wszystkie automatycznie umieszczone przedmioty będą wyświetlane "w kolejności", nawet jeśli pozostawi to, które mogłyby zostać wypełnione przez późniejsze przedmioty.

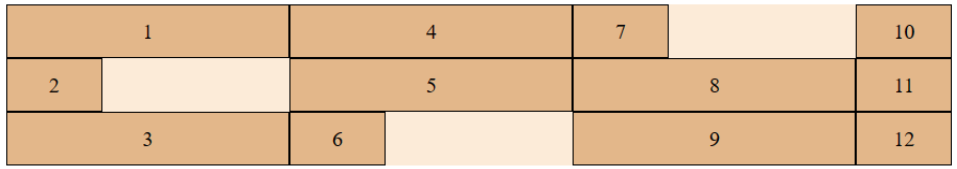
CSS Grid Layout

```
<section>
  <div>1</div>
  <div class="waskie">2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div class="waskie">6</div>
  <div class="waskie">7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
  <div class="waskie">11</div>
  <div>12</div>
</section>
```

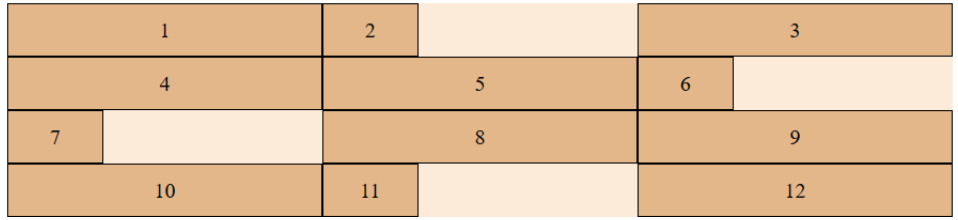
CSS Grid Layout

```css
section {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 1fr);
  background-color: antiquewhite;
  grid-auto-flow: column dense;
}

div {
  border: 1px solid black;
  background-color: burlywood;
  padding: 10px;
  text-align: center;
}
.waskie{
  width: 50px
}
```

`grid-auto-flow: column dense;`



`grid-auto-flow: row dense;`



`justify-content`:

Właściwość działa tylko wtedy, gdy łączna szerokość kolumn jest mniejsza niż szerokość kontenera siatki. Innymi słowy, potrzebujesz wolnego miejsca wzdłuż osi wiersza kontenera, aby wyrównać jego kolumny w lewo lub w prawo.

- start
- center
- end
- stretch
- space-between
- space-around
- space-evenly

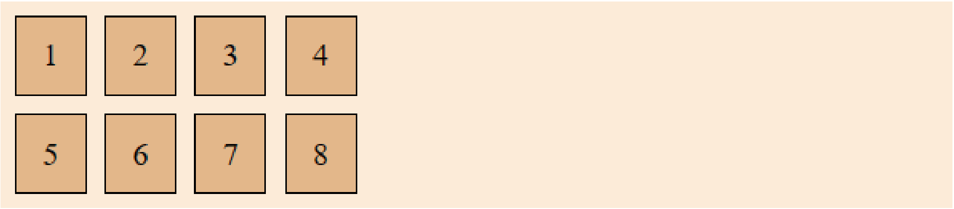
CSS Grid Layout

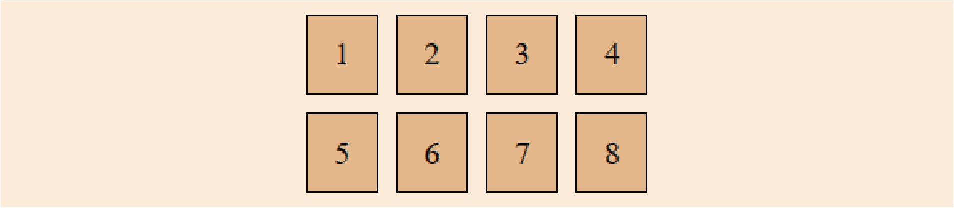
```html
<section>
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
</section>
```

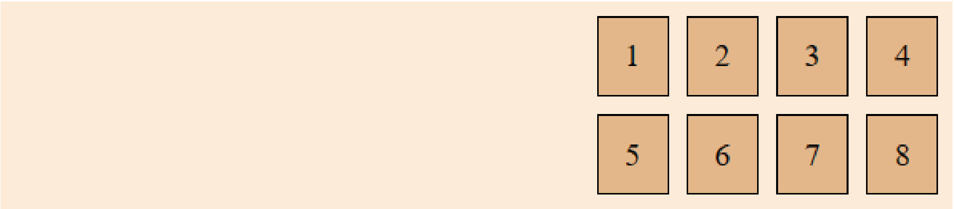
```css
section {
  display: grid;
  justify-content: start;
  grid-template-columns: repeat(4, 40px);
  background-color: antiquewhite;
  gap: 5px;
  padding: 5px;
}

div {
  border: 1px solid black;
  background-color: burlywood;
  padding: 10px;
  text-align: center;
  margin: 2px;
}
```

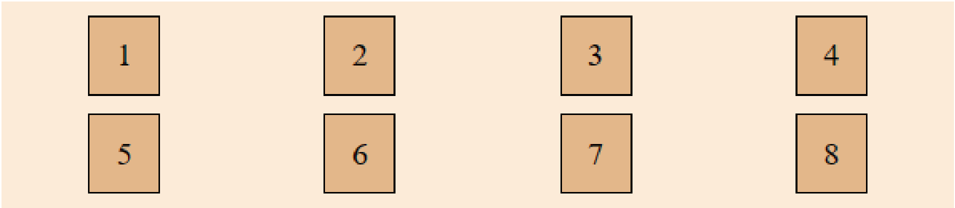
CSS Grid Layout

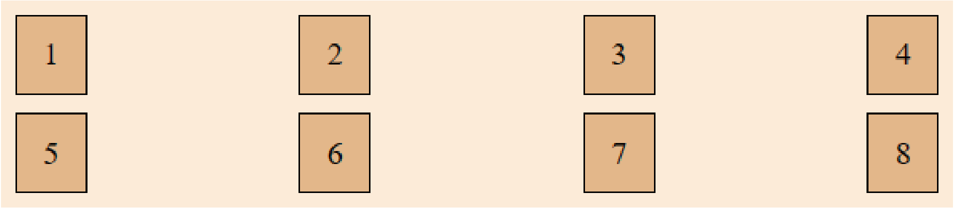
`justify-content:` start; center; end;

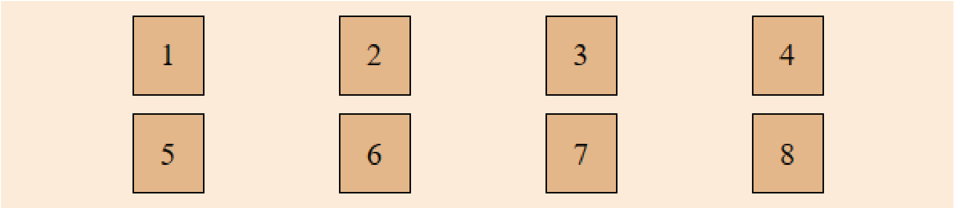












`justify-content:` space-between
space-around
space-evenly

CSS Grid Layout

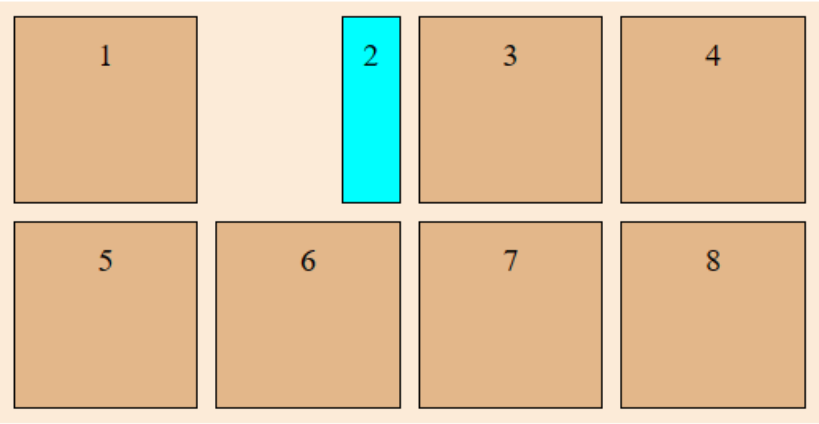
`justify-self: flex-start;`

Właściwość CSS justify-self określa sposób justowania pola wewnątrz kontenera wyrównania wzdłuż odpowiedniej osi.

```
/* Basic keywords */
justify-self: auto;
justify-self: normal;
justify-self: stretch;

/* Positional alignment */
justify-self: center; /* Pack item around the center */
justify-self: start; /* Pack item from the start */
justify-self: end; /* Pack item from the end */
justify-self: flex-start; /* Equivalent to 'start'. Note that justify-self is ignored in Flexbox layouts. */
justify-self: flex-end; /* Equivalent to 'end'. Note that justify-self is ignored in Flexbox layouts. */
justify-self: self-start;
justify-self: self-end;
justify-self: left; /* Pack item from the left */
justify-self: right; /* Pack item from the right */

/* Baseline alignment */
justify-self: baseline;
justify-self: first baseline;
justify-self: last baseline;

/* Overflow alignment (for positional alignment only) */
justify-self: safe center;
justify-self: unsafe center;

/* Global values */
justify-self: inherit;
justify-self: initial;
justify-self: revert;
justify-self: revert-layer;
justify-self: unset;
```

justify-self

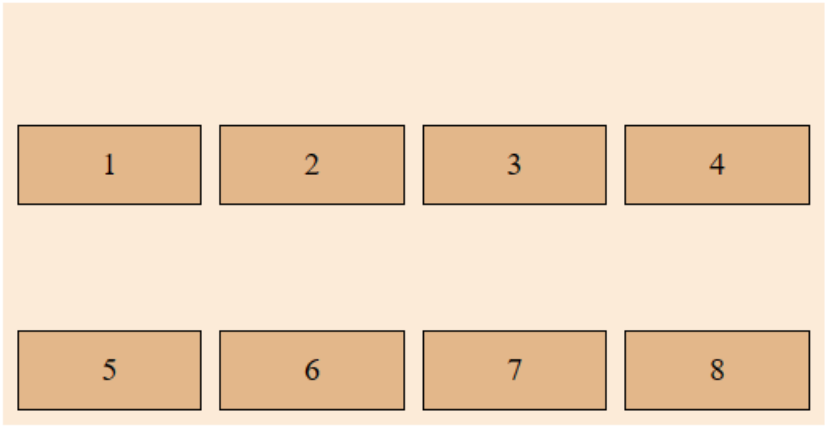
Wyrównanie wewnątrz komórki siatki. Obiekt nie wypełnia całej komórki, lecz zachowuje swoją szerokość dostosowaną do zawartości.




```
section {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-template-rows: repeat(2, 1fr);
  background-color: antiquewhite;
  gap: 5px;
  padding: 5px;
  width: 400px;
  height: 200px;
}

div {
  border: 1px solid black;
  background-color: burlywood;
  padding: 10px;
  text-align: center;
  margin: 2px;
}
.test{
background-color: aqua;
justify-self:right;
}
```

`align-items`:

Wyśrodkowany obiektów wewnątrz komórki w pionie




```
section {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-template-rows: repeat(2, 1fr);
  background-color: antiquewhite;
  gap: 5px;
  padding: 5px;
  width: 400px;
  height: 200px;
  align-items: flex-end;
}

div {
  border: 1px solid black;
  background-color: burlywood;
  padding: 10px;
  text-align: center;
  margin: 2px;
}
```