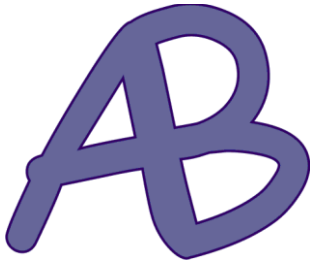


Media Query

Media Query

Reguła @media określa docelowe typy nośników (oddzielone przecinkami) zestawu instrukcji (rozdzielanych nawiasami klamrowymi). Nieprawidłowe instrukcje muszą być ignorowane. Konstrukcja @media zezwala na reguły arkusza stylów dla różnych nośników w tym samym arkuszu stylów:

```
@media print {
  body { font-size: 10pt }
}
@media screen {
  body { font-size: 13px }
}
@media screen, print {
  body { line-height: 1.2 }
}
```

Media Query

all - Nadaje się do wszystkich urządzeń.

braille - Przeznaczony do dotykowych urządzeń sprzężenia zwrotnego w alfabecie Braille'a.

embossed - Przeznaczony do stronicowanych drukarek brajlowskich

handheld - Przeznaczony do urządzeń przenośnych (zazwyczaj mały ekran, ograniczona przepustowość).

print - Przeznaczony do materiałów stronicowanych i dokumentów wyświetlanych na ekranie w trybie podglądu wydruku. Zapoznaj się z sekcją dotyczącą nośników stronicowanych, aby uzyskać informacje na temat problemów z formatowaniem, które są specyficzne dla multimediów stronicowanych.

projection - Przeznaczony do projekcji prezentacji, na przykład projektorów. Zapoznaj się z sekcją dotyczącą nośników stronicowanych, aby uzyskać informacje na temat problemów z formatowaniem, które są specyficzne dla multimediów stronicowanych.

screen - Przeznaczony przede wszystkim do kolorowych ekranów komputerowych.

speech - Przeznaczony do syntezatorów mowy. Uwaga: CSS2 miał podobny typ nośnika o nazwie "dźwiękowy" w tym celu. Szczegółowe informacje można znaleźć w dodatku do dźwiękowych arkuszy stylów.

Media Query

Obecnie istnieją dwa sposoby określania zależności między mediami dla arkuszy stylów:

- Określ nośnik docelowy z arkusza stylów z @media lub @import regułami

```
@import url("fancyfonts.css") screen;
@media print {
  /* style sheet for print goes here */
}
```

- Określ nośnik docelowy w języku dokumentu. Na przykład w HTML 4 atrybut "media" w elemencie LINK określa nośnik docelowy zewnętrznego arkusza stylów:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<HTML>
  <HEAD>
    <TITLE>Link to a target medium</TITLE>
    <LINK REL="stylesheet" TYPE="text/css" MEDIA="print, handheld" HREF="foo.css">
  </HEAD>
  <BODY>
    <P>The body...
  </BODY>
</HTML>
```

Media Query

Zapytanie mediów składa się ze standardowego typu lub grupy medium, po którym następuje słowo kluczowe "and" (ang. i), a następnie w nawiasie cecha medium (ang. media feature), określająca wymagane możliwości urządzenia w ramach podanego na początku mediów ogólnego.

```
@media screen and (max-width: 800px) and (min-width: 400px){
  /* jeśli szerokość jest mniejsza od 800px i większa  od 400px */
  body{ background-color: lightblue; }
}
```

Media Query

Właściwości dla reguły @media.	
width	określenie wartości szerokości okna przeglądarki internetowej
height	określenie wartości wysokości okna przeglądarki internetowej
device-width	określenie wartości rozdzielczości ekranu urządzenia (szerokość)
device-height	określenie wartości rozdzielczości ekranu urządzenia (wysokość)
color	określenie liczby bitów na kolor lub określenie czy urządzenie posiada kolorowy ekran
color-index	określenie wartości głębi kolorów, które obsługuje dane urządzenie
aspect-ratio	określenie wartości proporcji szerokości do wysokości okna przeglądarki internetowej
device-aspect-ratio	określenie wartości proporcji szerokości do wysokości rozdzielczości ekranu urządzenia
grid	określenie urządzenia z ograniczonymi możliwościami wyświetlania
monochrome	określenie liczby bitów na piksel w urządzeniach monochromatycznych, jednokolorowych
orientation	określenie orientacji pionowej lub poziomej urządzenia
resolution	określenie wartości gęstości pikseli dla danego urządzenia
scan	określenie czy urządzenie posiada skanowanie obrazu progresywne czy międzyliniowe

Media Query

Operatory logiczne dla reguł **@media.**

and	operator **" i "**, służy do tworzenia bardziej precyzyjnych warunków w regule **@media**
„przecinek”	Operator „przecinek” oznacza **" lub "**, służy do tworzenia bardziej precyzyjnych warunków w regule **@media**
not	operator **" negacji "**, służy do tworzenia bardziej precyzyjnych warunków w regule **@media**
only	operator przeznaczony dla starszych przeglądarek internetowych

Media Query

Atrybuty przydatne do tworzenia warunków

Inne atrybuty, które warto znać to:

min-width	Warunki obowiązują w każdej przeglądarce, której szerokość przekracza wartość podaną w zapytaniu.
max-width	Reguły dotyczą każdej przeglądarki, której szerokość jest poniżej wartości podanej w zapytaniu.
min-height	Reguły będą obowiązywać w każdej przeglądarce, której wysokość przekroczy wartość podaną w zapytaniu.
max-height	Warunki obowiązują w każdej przeglądarce, której wysokość jest poniżej wartości podanej w zapytaniu.
orientation=portrait	Reguły będą w każdej przeglądarce, której wysokość jest równa szerokości lub większa.
orientation=landscape	Reguły obowiązują w każdej przeglądarce, której szerokość jest większa niż wysokość.

viewport

Samo dodanie warunków @media nie sprawi, że nasza strona będzie się dobrze skalować. Trzeba na stronie pomiędzy znacznikami <head></head> dodać odpowiednią dyrektywę:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

lub prościej

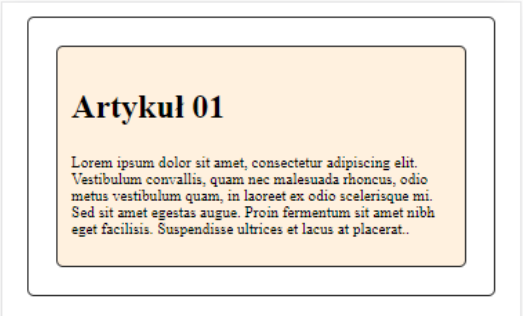
```
<meta name="viewport" content="width=device-width" />
```

Dzięki, której przeglądarka będzie wiedział, że ma zwracać uwagę na szerokość okna w przypadku wyświetlania strony.

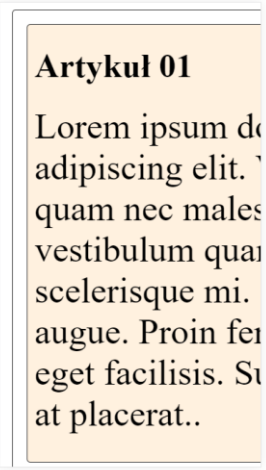
viewport

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flex</title>
    <link rel="stylesheet" type="text/css" href="styl01.css" />
  </head>
  <body>
    <section>
      <article>
        <h2>Artykuł 01</h2>
        <p>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum
          convallis, quam nec malesuada rhoncus, odio metus vestibulum quam, in
          laoreet ex odio scelerisque mi. Sed sit amet egestas augue. Proin
          fermentum sit amet nibh eget facilisis. Suspendisse ultrices et lacus
          at placerat..
        </p>
      </article>
    </section>
  </body>
</html>
```

```css
section,
article {
  margin: 10px;
  padding: 10px;
  border: 1px solid rgb(47, 47, 47);
  border-radius: 5px;
}
p {
  font-size: 2vw;
}
article {
  background-color: rgb(255, 241, 223);
}
```



Bez tagu viewport



viewport

Viewport to tzw. obszar renderowania treści w wirtualnym oknie przeglądarki wyświetlanym na ekranie. Mówiąc najprościej, to obszar przeglądarki, w którym widzimy stronę www.

Optymalizacja strony pod urządzenia mobilne to obecnie kluczowy proces dla Front-End Developerów. Podstawą dla uzyskania responsywności jest umieszczenie meta tagu viewport (w sekcji head dokumentu HTML), który daje przeglądarce informacje, w jaki sposób strona ma być wyświetlana na urządzeniach mobilnych.

Znacznik ten pozwala ustawić opcje skalowania strony, w tym także domyślne przybliżenie (zoom). Podana wartość 1 dla `initial-scale` wyświetla ją bez skalowania.

Dla dyrektywy width ustawiona wartość `device-width` powoduje, że szerokość obszaru wyświetlania jest równa szerokości ekranu urządzenia, a więc witryna jest rozciągnięta w 100%.

Aby określić szerokość i wysokość wyświetlanego obszaru strony możemy podać dla width i height wartość liczbową w pikselach.

viewport

Viewport to tzw. obszar renderowania treści w wirtualnym oknie przeglądarki wyświetlanym na ekranie. Mówiąc najprościej, to obszar przeglądarki, w którym widzimy stronę www.

Optymalizacja strony pod urządzenia mobilne to obecnie kluczowy proces dla Front-End Developerów. Podstawą dla uzyskania responsywności jest umieszczenie meta tagu viewport (w sekcji head dokumentu HTML), który daje przeglądarce informacje, w jaki sposób strona ma być wyświetlana na urządzeniach mobilnych.

Znacznik ten pozwala ustawić opcje skalowania strony, w tym także domyślne przybliżenie (zoom). Podana wartość 1 dla `initial-scale` wyświetla ją bez skalowania.

Dla dyrektywy width ustawiona wartość `device-width` powoduje, że szerokość obszaru wyświetlania jest równa szerokości ekranu urządzenia, a więc witryna jest rozciągnięta w 100%.

Aby określić szerokość i wysokość wyświetlanego obszaru strony możemy podać dla width i height wartość liczbową w pikselach.

viewport

Jednostki viewport CSS

Aby uzyskać dynamicznie zmieniające się rozmiary elementów na stronie względem okna przeglądarki, należy ustawić im rozmiar za pomocą jednostek tzw. Viewport Units, które reprezentują procent aktualnych wymiarów obszaru roboczego ekranu:

vw – (viewport width) procentowa szerokość – względem szerokości okna przeglądarki;

vh – (viewport height) procentowa wysokość – względem wysokości okna przeglądarki;

vmin – (viewport minimum) mniejsza z wartości vw lub vh;

vmax – (viewport maximum) większa z wartości vw lub vh.

viewport

Jednostki viewport CSS

Aby uzyskać dynamicznie zmieniające się rozmiary elementów na stronie względem okna przeglądarki, należy ustawić im rozmiar za pomocą jednostek tzw. Viewport Units, które reprezentują procent aktualnych wymiarów obszaru roboczego ekranu:

vw – (viewport width) procentowa szerokość – względem szerokości okna przeglądarki;

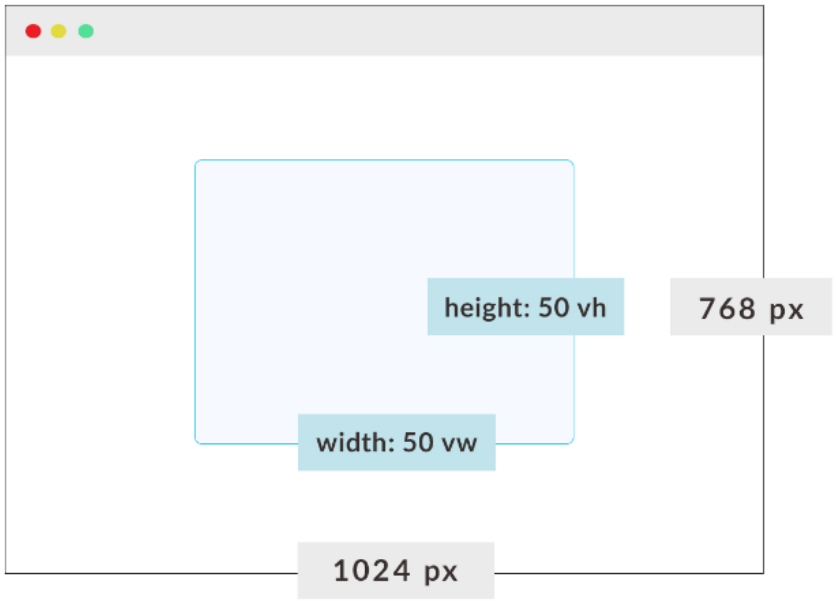
vh – (viewport height) procentowa wysokość – względem wysokości okna przeglądarki;

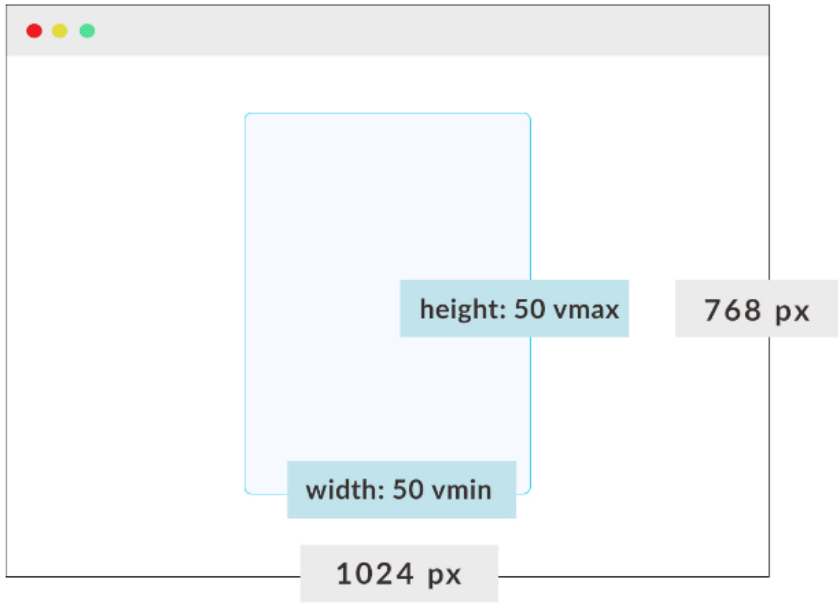
vmin – (viewport minimum) mniejsza z wartości vw lub vh;

vmax – (viewport maximum) większa z wartości vw lub vh.

viewport

Źródło: https://ui2web.com/frontend/jednostki-viewport-na-czym-polegaja/





width: 50vw – element o wysokości połowy wysokości aktualnego okna przeglądarki
height: 50vh – element o wysokości połowy wysokości aktualnego okna przeglądarki

width: 50vmin – element o szerokości połowy z aktualnie większej wartości – wysokości lub szerokości

height: 50vmax – element o wysokości połowy z aktualnie mniejszej wartości – wysokości lub szerokości

Funkcja CSS calc()

Funkcja calc() przy zastosowaniu jednostek viewport i stałych jednostek zapobiega niekontrolowanemu zmniejszaniu rozmiarów elementu. Dzięki niej możemy ustanowić podstawową wartość tekstu na 14px i dodać do niej 2 jednostki vw.

```
.main-text {
  font-size: calc(14px + 2vw);
}
```

Aby zapobiec znacznemu powiększeniu tekstu na dużych ekranach, musimy zastosować standardowe Media Queries:

```
@media (min-width: 1600px) {
 .main-text {
   font-size:40px;
 }
}
```

Przykład

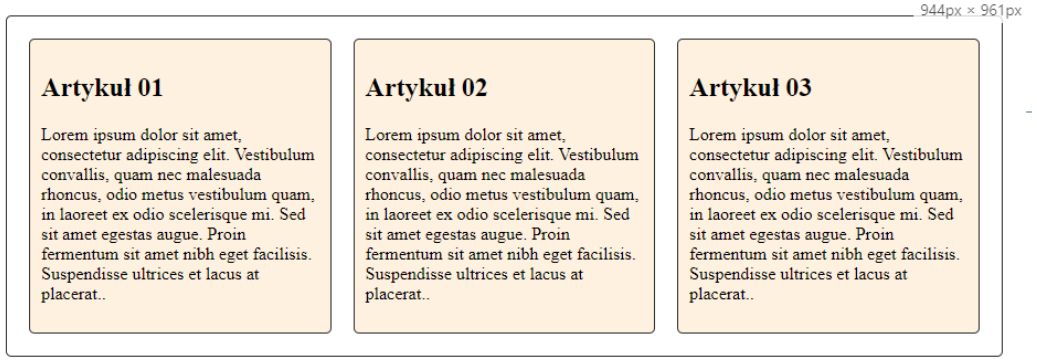
```
section,
article {
  margin: 10px;
  padding: 10px;
  border: 1px solid rgb(47, 47, 47);
  border-radius: 5px;
}

article {
  background-color: rgb(255, 241, 223);
}

body {
  display: flex;
  flex-wrap: wrap;
}
section {
  display: flex;
  flex-wrap: wrap;
}

@media screen and  (min-width: 800px){

  article {
    max-width: calc(33vw - 60px);
  }
}
```



Funkcja calc( )

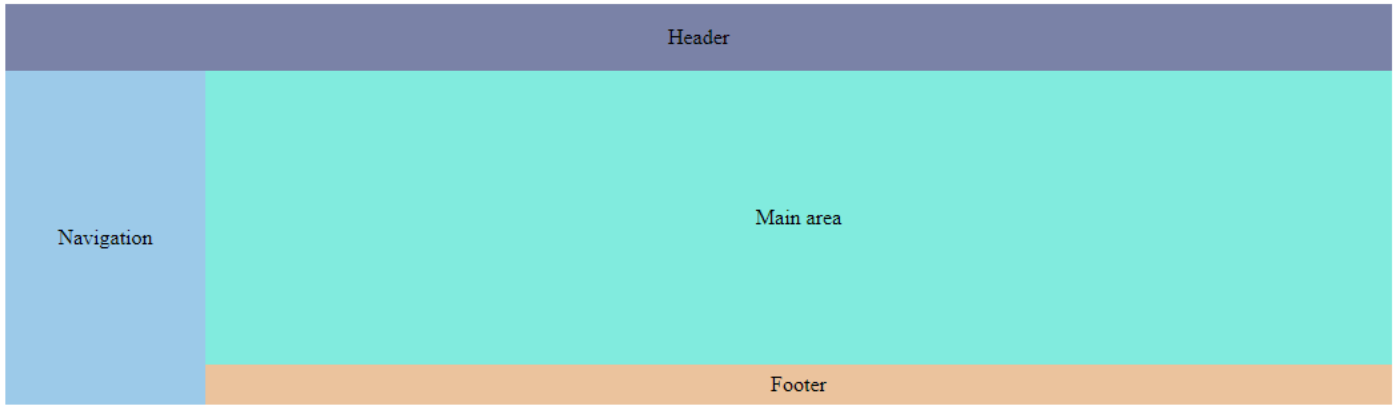
CSS ma specjalną funkcję do wykonywania podstawowej matematyki.

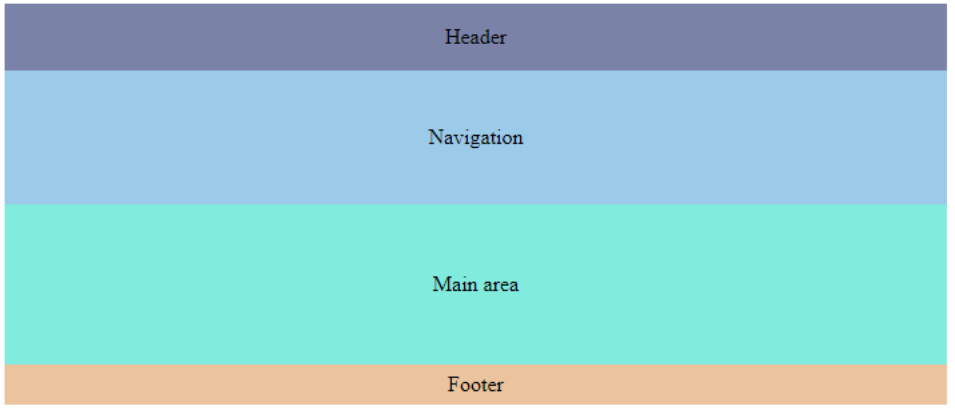
Przykłady:

```
font-size: calc(3vw + 2px);
width: calc(100% - 20px);
height: calc(100vh - 20px);
padding: calc(1vw + 5px);
border-radius: 15px calc(15px / 3) 4px 2px;
```

Przykład 2

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" type="text/css"
          href="styl_grid.css">
    <title></title>
  </head>
  <body>
    <section id="page">
      <header>Header</header>
      <nav>Navigation</nav>
      <main>Main area</main>
      <footer>Footer</footer>
    </section>
  </body>
</html>
```





Przykład 2

```css
#page {
  display: grid;
  width: 100%;
  height: 500px;
  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;
}
```

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

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

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

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

```css
@media screen and (max-width: 800px) {
  #page {
    grid-template-areas:
      "head"
      "nav"
      "main"
      "foot";
    grid-template-rows: 50px 100px 1fr 30px;
    grid-template-columns: 100%;
  }
}
```