

Akademia Górniczo-Hutnicza im. St. Staszica w Krakowie

Responsywność stron www

Tworzenie stron www

Tomasz Bartuś

Wyłącznie do użytku wewnętrznego AGH.

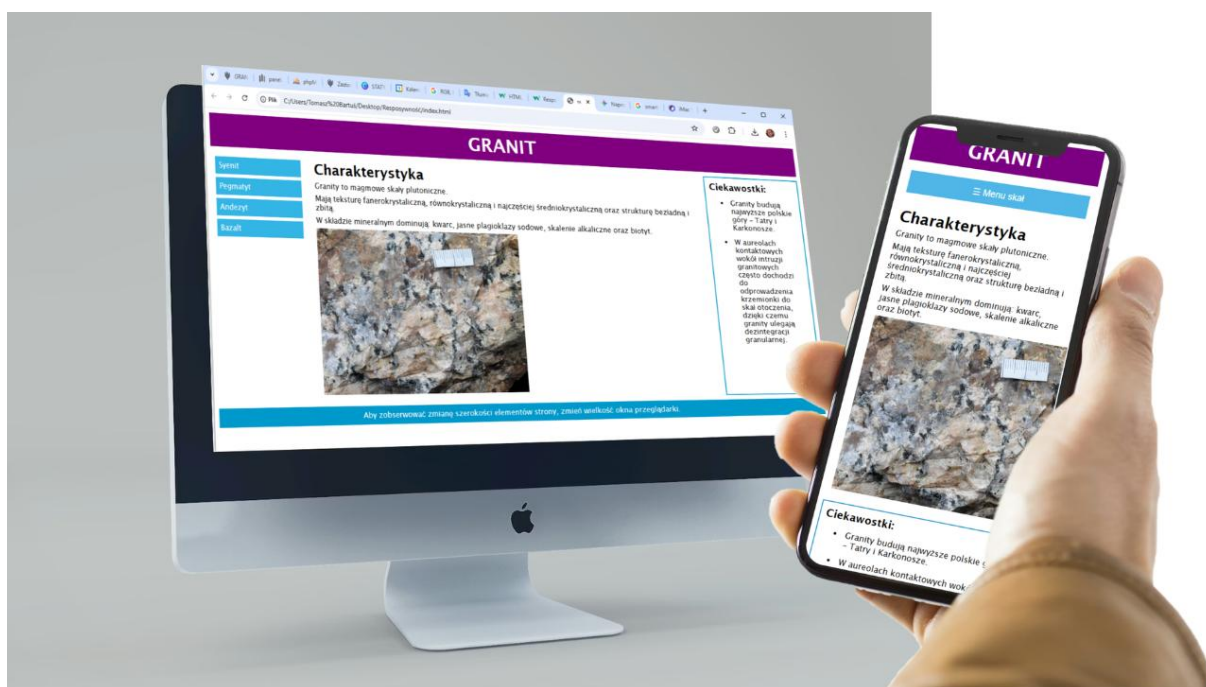
<http://home.agh.edu.pl/~bartus>

2026-04-10

Wprowadzenie

Jedną z najważniejszych cech XXI wieku jest powszechny dostęp do informacji. Internet opanował świat. Dokonujemy zakupów, przeglądamy informacje bieżące, jesteśmy aktywni w socialmediach, wyszukujemy informacji na rozmaite tematy. Korzystamy z Internetu na bardzo różnych urządzeniach – począwszy od mobilnych (smartfonów, tabletów, notebooków), poprzez komputery osobiste, a skończywszy na wielkoformatowych telewizorach, tablicach multimedialnych itp. Urządzenia posiadają bardzo zróżnicowane rozdzielczości ekranów. Jednym z podstawowych problemów jakie pojawiły się w związku z tą różnorodnością był inny wygląd stron internetowych oglądanych na urządzeniach o małych, średnich i wielkich ekranach. Zaistniała konieczność opracowania technologii, która płynnie dostosowywałaby wygląd strony www do rozdzielczości ekranu urządzenia, na którym tę stronę przeglądamy. Odpowiedzią na ten problem jest Responsive Web Design.

W ramach ćwiczenia utworzymy projekt prostej responsywnej strony www (Ryc. 1).



Ryc. 1. Projekt responsywnej strony www

Responsywność

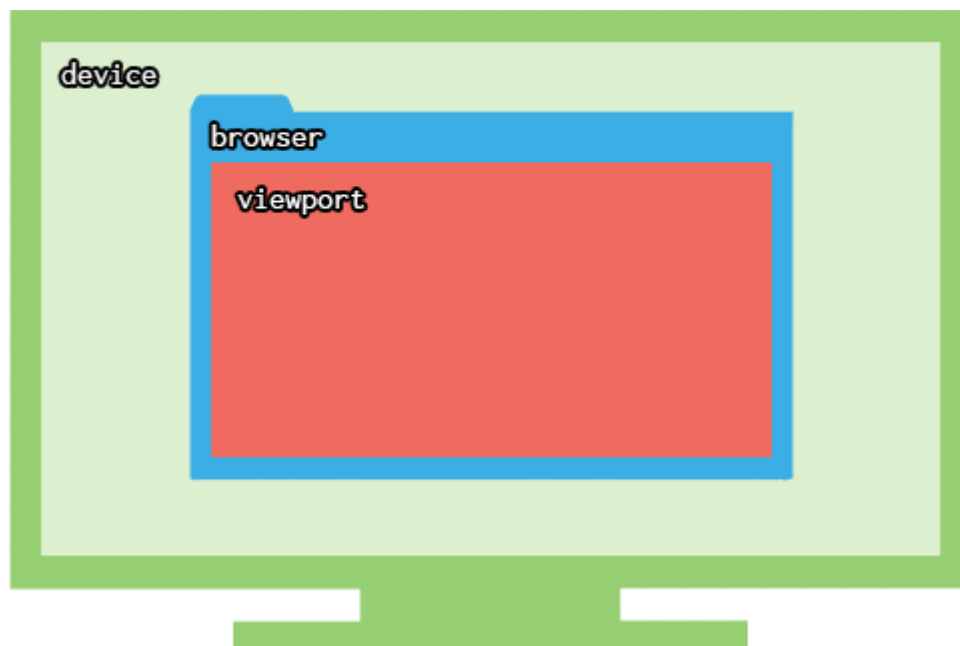
Responsive Web Design (RWD) to projektowanie stron internetowych, których układ automatycznie dostosowuje się do wielkości ekranu urządzenia. Wyświetlane na ekranie treści skalują się aby dobrze wyglądać na każdym urządzeniu. Strony responsywne cechuje uniwersalność, czego dowodem jest ich atrakcyjny wygląd zarówno na komputerze desktopowym, laptopie, tablecie czy smartfonie (Ryc. 2).



Ryc. 2. Zmienny wygląd strony internetowej w zależności od urządzenia

1. Viewport

Viewport (pole widzenia) to widoczny dla użytkownika obszar strony internetowej (Ryc. 3). Różni się on w zależności od urządzenia i będzie mniejszy na telefonie komórkowym niż na ekranie komputera.



Ryc. 3. Viewport to zakres widzenia strony www, który jest zależny od rozmiarów wykorzystywanych urządzeń

Przed epoką tabletów i telefonów komórkowych strony internetowe były projektowane tylko na ekrany komputerów. Miały wtedy stały rozmiar. Wszystko zmieniło się po ekspansji tabletów i telefonów komórkowych. Strony internetowe o stałym rozmiarze były zbyt duże aby zmieścić się w *viewporcie* urządzeń mobilnych. Aby to naprawić, przeglą-

darki na tych urządzeniach zmniejszyły całą stronę internetową aby dopasować ją do ekranu.

1.1. Ustawianie Viewport

HTML 5.0 wprowadził metodę pozwalającą projektantom stron internetowych przejąć kontrolę nad polem widzenia za pomocą znacznika `<meta>`.

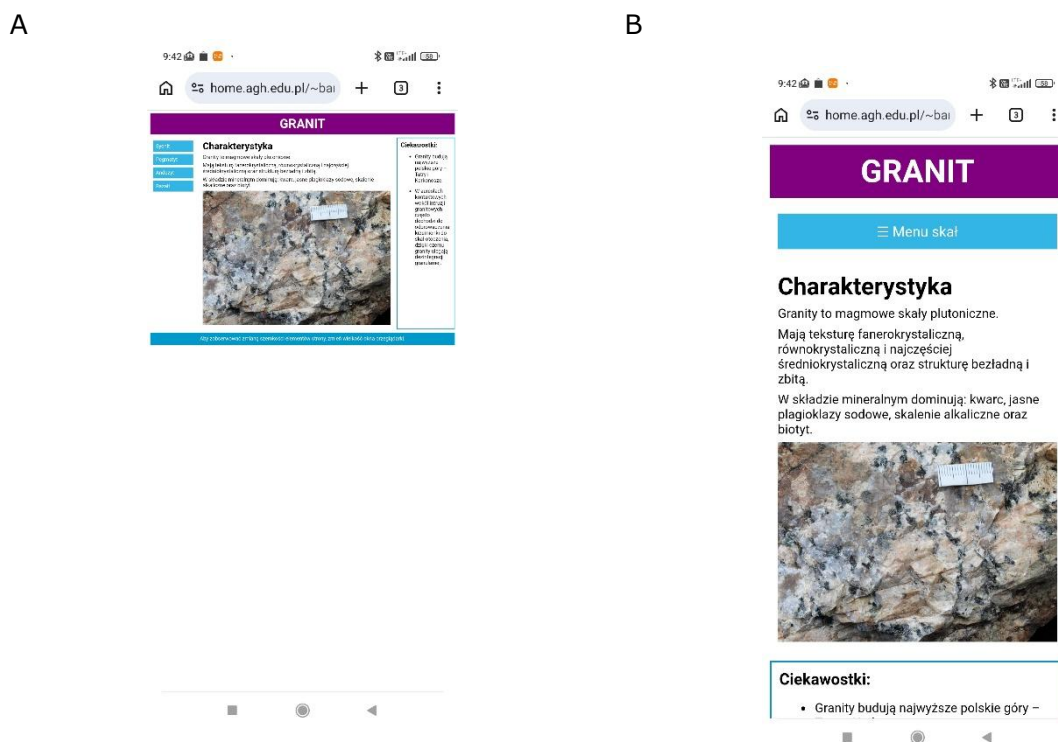
W sekcji `<head>` wszystkich stron internetowych należy uwzględnić następujący element `<meta>`:

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

Zawiera on instrukcje dla przeglądarki dotyczące sposobu kontrolowania wymiarów i skalowania strony.

- `width=device-width` ustawia szerokość strony tak aby odpowiadała szerokości ekranu urządzenia (która będzie się różnić w zależności od urządzenia).
- `initial-scale=1.0` gdy strona jest ładowana po raz pierwszy przez przeglądarkę ustawia początkowy poziom powiększenia.

Ryc. 4 przedstawia realizowany projekt strony internetowej bez znacznika `meta viewport` i tej samej strony internetowej ze znacznikiem `meta viewport`:



Ryc. 4. Projekt strony www przeglądany na smartfonie bez deklaracji viewport (A) i z deklaracją (B)

1.2. Rozmiar zawartości do obszaru widoku

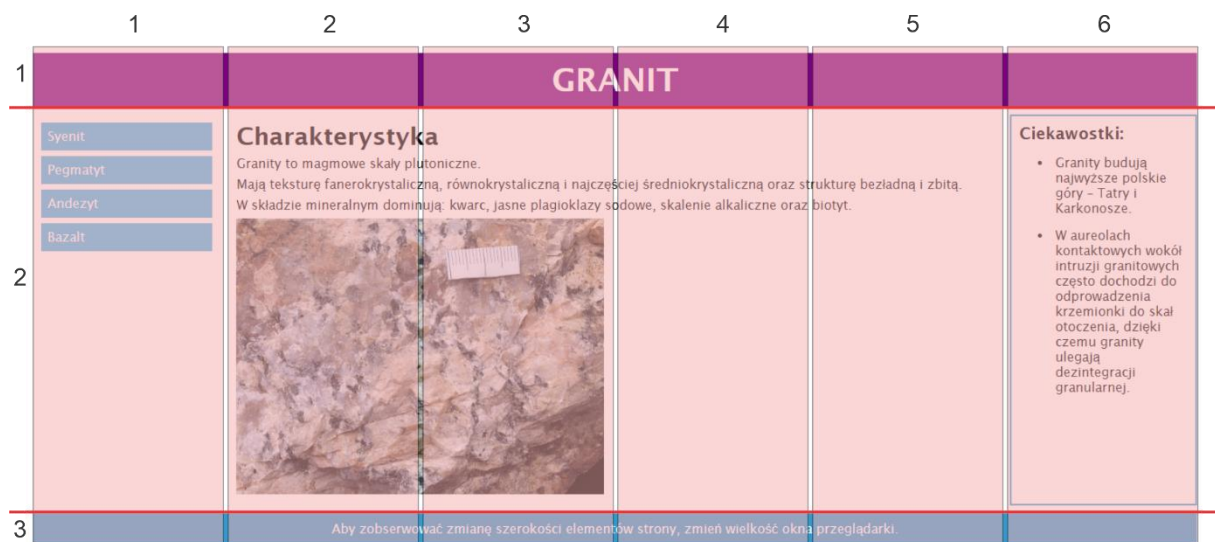
Zarówno na komputerach stacjonarnych, jak i urządzeniach mobilnych użytkownicy są przyzwyczajeni do przewijania stron internetowych w pionie — ale nie w poziomie. Jeśli więc użytkownik jest zmuszony przewijać stronę w poziomie lub ją pomniejszyć aby zobaczyć całą stronę internetową, skutkuje to irytacją.

Zasady, które należy przestrzegać:

1. NIE używaj dużych elementów o stałej szerokości — na przykład, jeśli obraz ma szerokość większą niż *viewport*, powoduje to przewijanie *viewport* w poziomie. Pamiętaj, aby dostosować tę zawartość tak aby pasowała do szerokości *viewport*.
2. Ponieważ wymiary ekranu i szerokości w pikselach CSS różnią się w zależności od urządzenia, zawartość, aby dobrze się renderowała, nie powinna polegać na jednej konkretnej szerokości *viewport*.
3. Użyj zapytań o media CSS aby zastosować różne style dla małych i dużych ekranów — ustawienie dużych bezwzględnych szerokości CSS dla elementów strony spowoduje, że elementy będą zbyt szerokie dla mniejszych urządzeń. Zamiast tego rozważ użycie względnych wartości szerokości, takich jak `width: 100%`. Uważaj również na używanie dużych wartości pozycjonowania bezwzględnego. Może to spowodować, że na małych urządzeniach element znajdzie się poza *viewport*.

2. Budowanie Grid-View

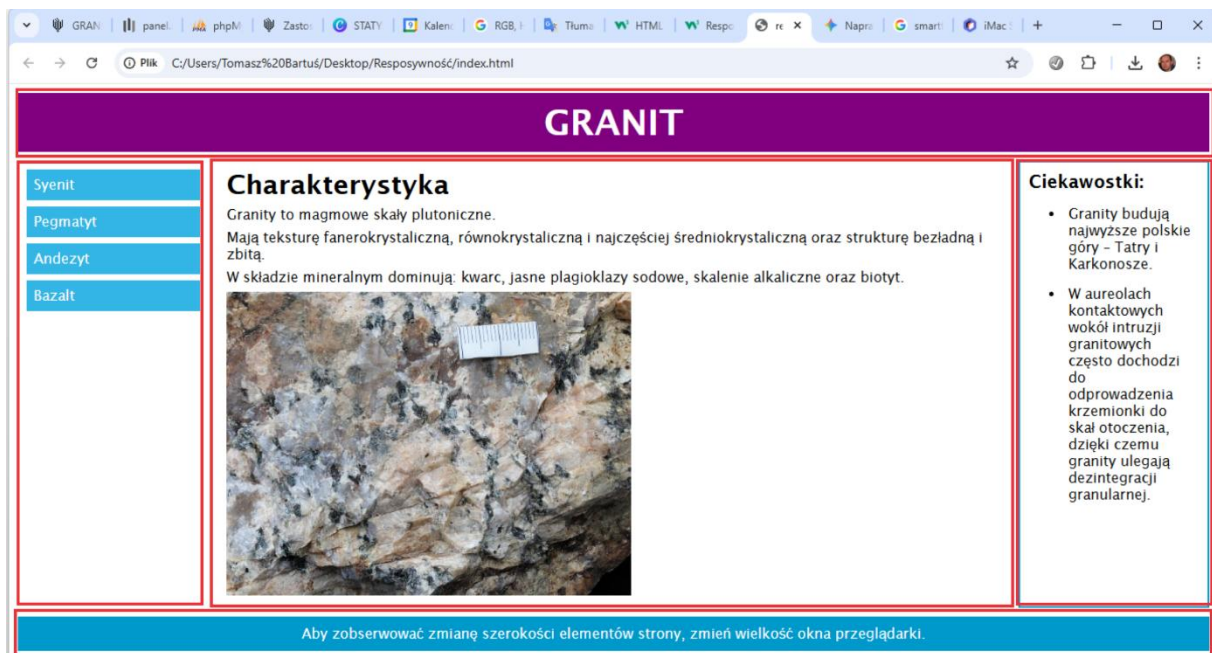
Wiele stron internetowych opiera się na **Grid-View**. Oznacza to, że strona jest podzielona na wiersze i kolumny (Ryc. 5). Responsywny *Grid-View* często ma **6** lub **12** kolumn i będzie się kurczył i rozszerzał w miarę zmiany rozmiaru okna przeglądarki. Używanie *Grid-View* jest bardzo pomocne podczas projektowania stron internetowych. Ułatwia umieszczanie elementów na stronie.



Ryc. 5. Podział viewport projektu strony internetowej za pomocą siatki na 6 kolumn i 3 wiersze

2.1. Tworzenie *Grid-View*

Każdy element na stronie HTML jest traktowany przez przeglądarkę jak prostokątne pudełko (Ryc. 6). Domyślnie jednak sposób obliczania rozmiaru każdego z pudełek bywa mylący.



Ryc. 6. Model pudełkowy projektu strony www

Istnieją dwa główne modele obliczania szerokości bloków (pudełek) (właściwość `block-sizing`):

- **content-box (domyślny sposób renderowania pudełek przez przeglądarki),**

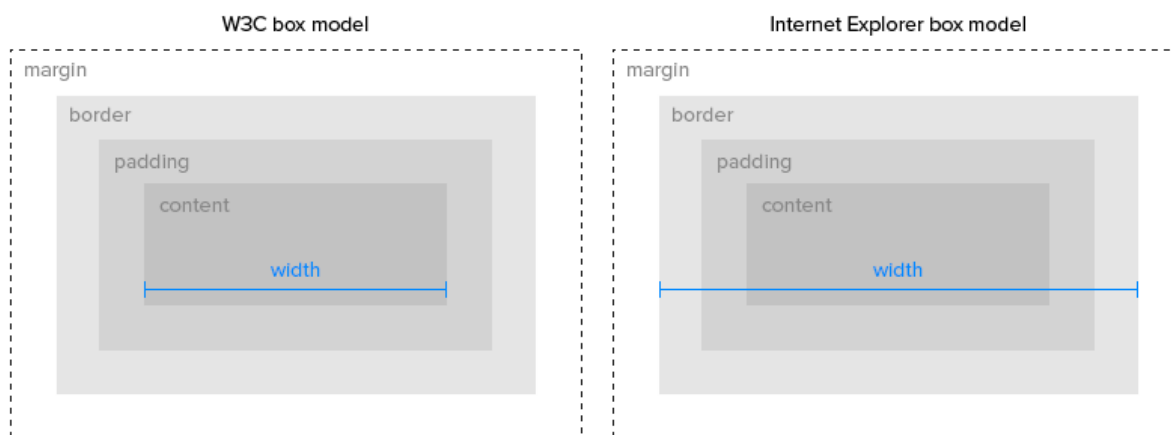
Jeśli ustawisz elementowi:

```
width: 100px;
padding: 20px;
border: 5px;
```

to jego rzeczywista szerokość na ekranie wyniesie **150px** (100 + 20 + 20 + 5 + 5). W efekcie tego elementy często "wystają" poza ekran lub niszczą układ grida, bo ich rozmiar końcowy robi się większy niż początkowo zadeklarowany (Ryc. 7).

- **border-box (zalecany w projektach RWD)**

Jeśli te same właściwości pudełka ustawisz przy `box-sizing: border-box`, element będzie miał dokładnie **100px** szerokości (Ryc. 7). Przeglądarka "wciśnie" margines wewnętrzny (`padding`) i ramkę (`border`) do środka, odpowiednio zwężając miejsce na treść. W efekcie mamy pełną kontrolę nad rozmiarem. Jeśli zechcemy, aby element miał 50% szerokości ekranu, to będzie miał dokładnie tyle, nawet jeśli dodamy mu grubą ramkę.



Ryc. 7. Sposób obliczania szerokości content w modelach pudełkowych W3C (`box-sizing: border-box;`) oraz Internet Explorer (`box-sizing: content-box;`)

W związku z tym, na samym początku budowania naszego projektu, w pliku CSS dodajmy tzw. **reset pudełkowy**.

2.1.1. Utwórz plik stylów.

2.1.2. Upewnij się, że wszystkie elementy HTML mają właściwość `box-sizing` ustawioną na `border-box`. Dzięki temu `padding` i `border` będą uwzględnione w całkowitej szerokości i wysokości elementów.

2.1.3. Dodaj poniższy kod na początku pliku CSS:

```
*{
  margin: 0;
  box-sizing: border-box;
}
```

}

2.2. HTML

2.2.1. Utwórz plik HTML, dołącz do niego utworzony plik styli.

2.2.2. W sekcji <body> utwórz kontener siatki z pięcioma blokami (`item1`, `item2`, `item3`, `item4` i `item5`).

```
<div class="grid-container">

  <div class="item1">
    <h1>GRANIT</h1>
  </div>

  <div class="item2">
    <ul>
      <li>Syenit</li>
      <li>Pegmatyt</li>
      <li>Andezyt</li>
      <li>Bazalt</li>
    </ul>
  </div>

  <div class="item3">
    <h1>Charakterystyka</h1>
    <p>Granity to magmowe skały plutoniczne.</p>
    <p>Mają teksturę fanerokrystaliczną, równokrystaliczną i najczęściej
średniokrystaliczną oraz strukturę bezładną i zbitą.</p>
    <p>W składzie mineralnym dominują: kwarc, jasne plagioklasy sodowe,
skalenie alkaliczne oraz biotyt.</p>
    
  </div>

  <div class="item4">
    <h2>Ciekawostki:</h2>
    <ul>
      <li>Granity budują najwyższe polskie góry - Tatry i Karkonosze.</li>
      <li>W aureolach kontaktowych wokół intruzji granitowych często docho-
dzi do odprowadzenia krzemionki do skał otoczenia, dzięki czemu granity
ulegają dezintegracji granularnej.</li>
    </ul>
  </div>

  <div class="item5">
    <p>Aby zobserwować zmianę szerokości elementów strony, zmień wielkość
okna przeglądarki.</p>
  </div>

</div>
```

2.3. CSS

Wróćmy do utworzonego pliku styli.

2.3.1. Dodaj do pliku CSS następujący kod:

```
*{
  margin: 0;
  box-sizing: border-box;
}
body{
  font-family: "Lucida Sans", sans-serif;
}
.grid-container{
  display: grid;
  grid-template-columns: repeat(6, 1fr);
  grid-template-areas:
    'header header header header header header'
    'menu main main main main right'
    'footer footer footer footer footer footer';
  gap: 10px;
  background-color: white;
  padding: 10px;
}
.grid-container > div{
  padding: 10px;
  font-size: 16px;
}
.item1{
  grid-area: header;
  background-color: purple;
  text-align: center;
  color: #fff;
}
.item1 > h1{
  font-size: 40px;
}
.item2{
  grid-area: menu;
}
.item2 ul{
  list-style-type: none;
  margin: 0;
  padding: 0;
}
.item2 li{
  padding: 8px;
  margin-bottom: 7px;
  background-color: #33b5e5;
  color: #fff;
}
.item2 li:hover{
  background-color: #0099cc;
}
.item3 {
  grid-area: main;
}
.item3 > h1{
  font-size: 30px;
  margin-bottom: 7px;
}
.item3 > p{
  margin-bottom: 7px;
}
.item4{
  grid-area: right;
  border: 2px solid #0099cc;
}
```

```

background-color: #fff;
padding: 15px;
color: #000;
}
.item4 > h2{
font-size: 20px;
padding-bottom: 10px;
}
.item4 li{
padding: 5px;
margin-bottom: 5px;
}
.item5{
grid-area: footer;
background-color: #0099cc;
color: #fff;
text-align: center;
}

```




Zanim przejdziemy do dalszej części ćwiczenia wytłumaczmy kilka elementów tego pliku.




Blok `.grid-container` dzieli projekt strony internetowej za pomocą siatki złożonej z 6 kolumn i 3 wierszy (zob. **Ryc. 5**). Dla kolejnych elementów siatki zastosujemy nazwy: (`header`, `menu`, `main`, `right` i `footer`).

Dodając do bloku `.grid-container` deklarację `repeat(6, 1fr)`, wymuszamy na przeglądarce, aby każda z 6 kolumn miała dokładnie taką samą szerokość, niezależnie od tego, ile tekstu jest w środku (np. w `menu`, czy `main`). Dzięki temu projekt będzie w pełni przewidywalny.

Za pomocą atrybutu `grid-area` definiujemy przynależność każdego bloku do odpowiedniego elementu siatki: (`item1: header`, `item2: menu`, `item3: main`, `item4: right`, `item5: footer`).

Sprawdźmy jeszcze co oznacza zapis CSS `.item1 > h1`.

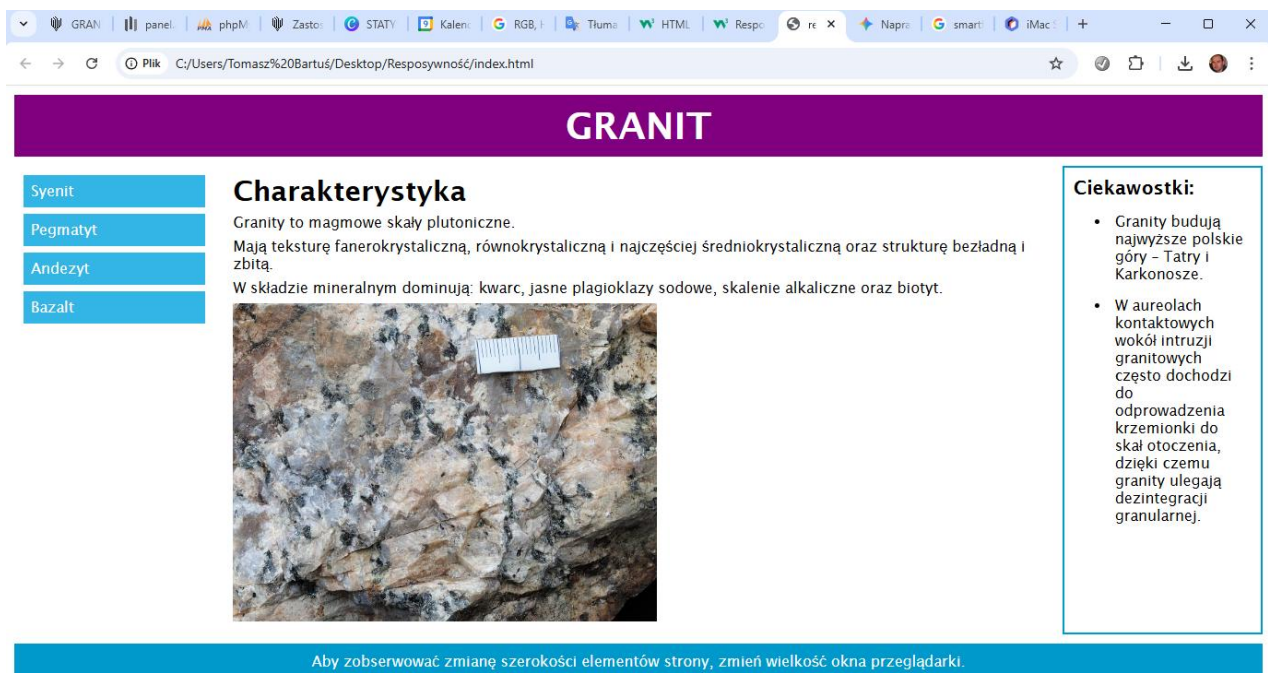
	<code>.item1 > h1 { ... }</code>	
	<code><div class="item1"> <h1>Pasuje</h1> </div></code>	✓ Pasuje
	<code><div class="item1"> <div> <h1>Nie pasuje</h1> </div> </div></code>	✗ NIE pasuje (bo nie jest bezpośrednim dzieckiem <code>.item1</code>)

	<code>.item1 h1 { ... }</code>	
	<code><div class="item1"> <h1>Pasuje</h1> </div></code>	✓ Pasuje
	<code><div class="item1"> <div> <h1>Też pasuje</h1> </div> </div></code>	✓ Również pasuje

Reasumując...

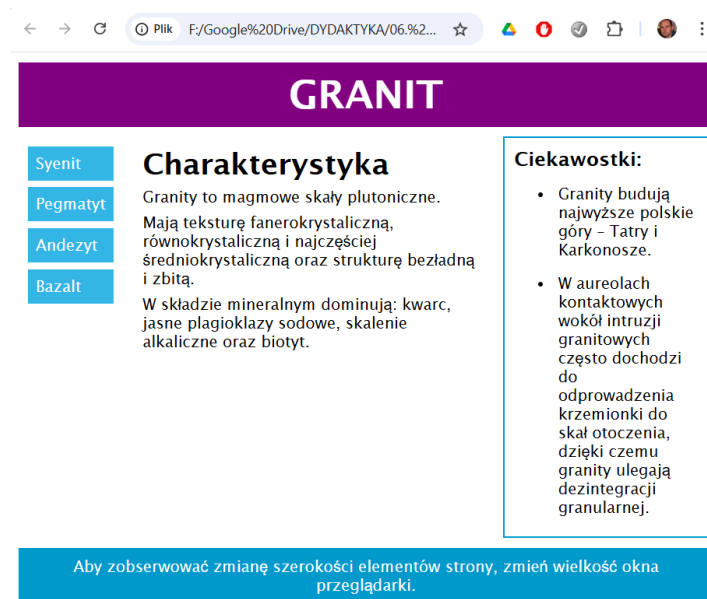
Selektor	Pasuje do bezpośrednich dzieci?	Pasuje do zagnieżdżonych głębiej?
<code>.item1 > h1</code>	✓ Tak	✗ Nie
<code>.item1 h1</code>	✓ Tak	✓ Tak

2.3.2. wyświetlaną w pełnej szerokości okna przeglądarki (Ryc. 8).



Ryc. 8. Strona internetowa w pełnej szerokości okna przeglądarki

2.3.3. Gdy zmienimy szerokość okna przeglądarki, zmienia się szerokość każdego bloku, zachowane są jednak proporcje zdefiniowane w CSS w atrybucie `grid-template-areas` (Ryc. 9).



Ryc. 9. Strona testowa po zmniejszeniu szerokości okna przeglądarki

Strona internetowa w poniższym przykładzie jest responsywna ale nie wygląda dobrze, gdy zmieniasz rozmiar okna przeglądarki na bardzo małą szerokość. W kolejnych rozdziałach dowiesz się jak to naprawić.

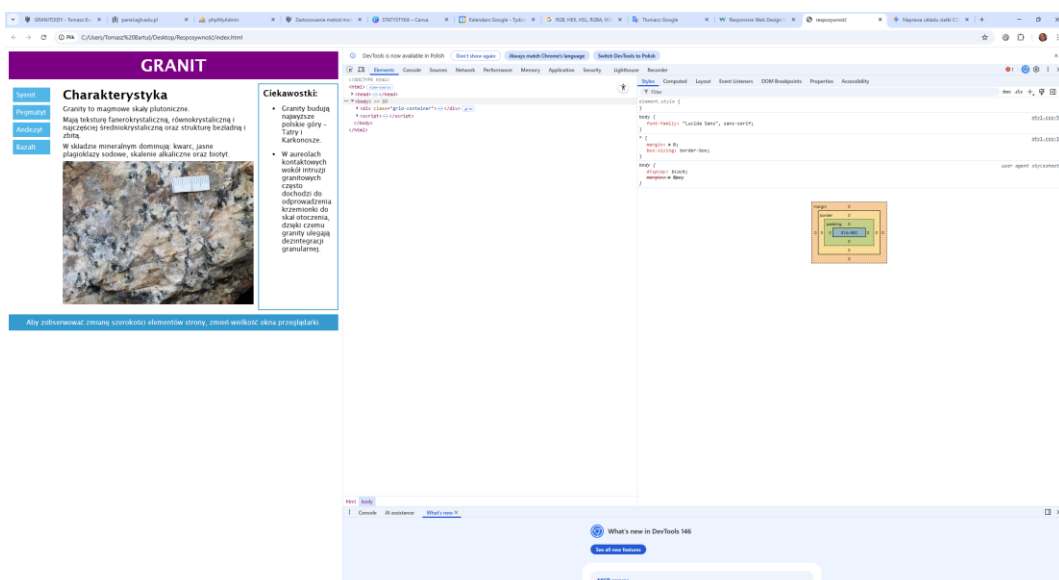
3. Wstęp do DevTools dla Web Developerów

Przetestujemy zachowanie strony www w profesjonalnym narzędziu developerskim.

2.3.2. Otwórz swoją stronę w przeglądarce Chrome (lub Firefox).

2.3.3. Naciśnij klawisz **F12** (lub prawy przycisk myszy > Zbadaj).

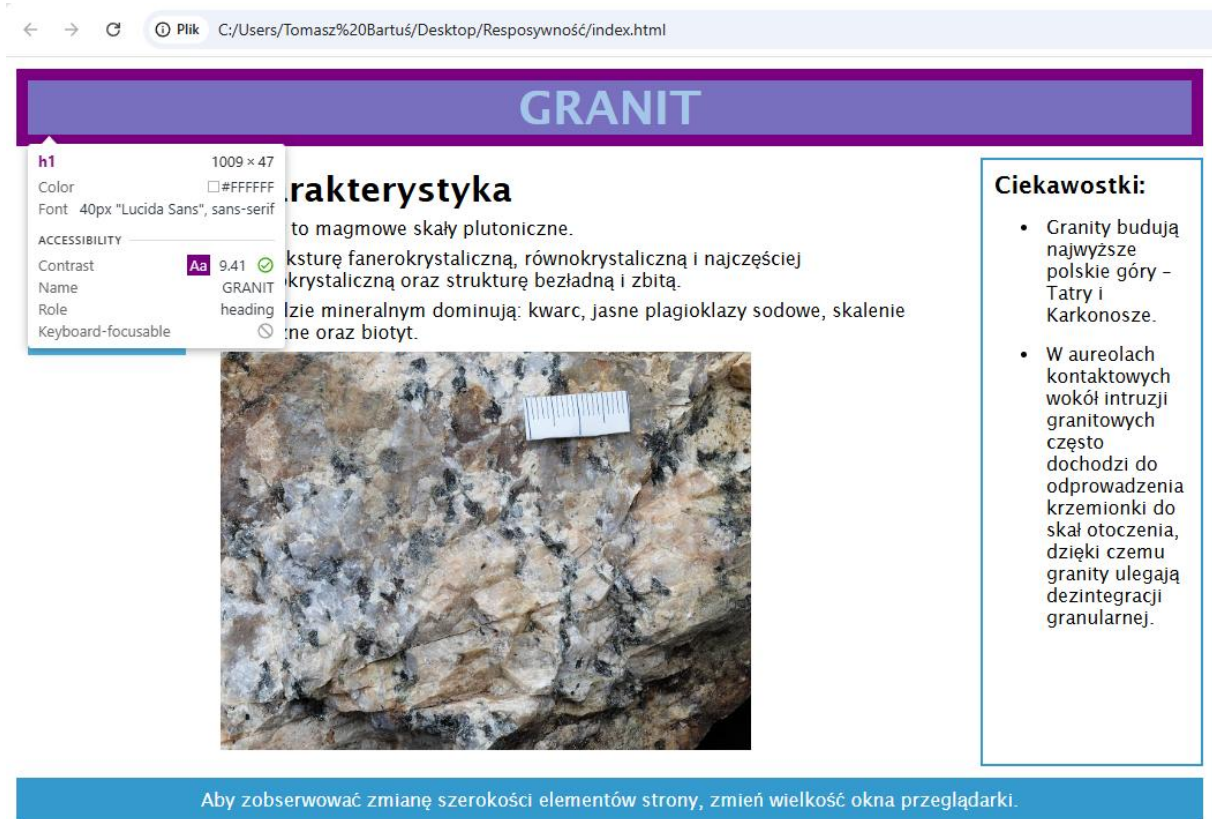
Strona otworzy się dodatku do przeglądarki internetowej DevTools (Ryc. 10).



Ryc. 10. Projekt strony www otwarty w Chrome Developer Tools

3.1. Inspektor elementów (*The Select Tool*)

3.1.1. W lewym górnym rogu *DevTools* znajdziesz ikonę strzałki w kwadracie (*Select an element in the page to inspect*). Kliknij tę ikonę, a następnie najedź na napis **GRANIT** na swojej stronie (*Ryc. 11*).



Ryc. 11. Widok projektu strony www z podstawowymi informacjami na temat wybranego elementu strony

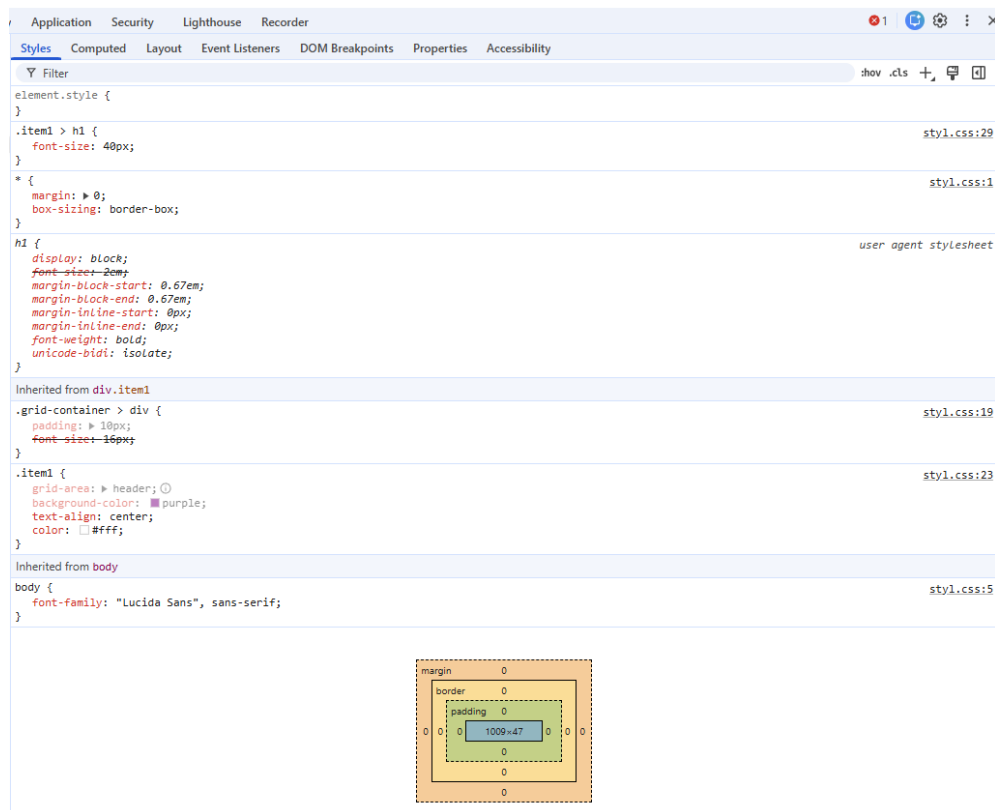
W zakładce *Elements* zobaczysz kod HTML tego nagłówka (*Ryc. 12*), a w panelu *Styles* po prawej stronie wszystkie reguły CSS, które go dotyczą (np. `background-color: purple`) (*Ryc. 13*).

```

Elements Console Sources Network Performance Memory
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <div class="grid-container">
      <div class="item1">
        <h1>GRANIT</h1>
      </div>
      <div class="item2">
      </div>
      <div class="item3">
      </div>
      <div class="item4">
      </div>
      <div class="item5">
      </div>
    </div>
    <script>
    </script>
  </body>
</html>

```

Ryc. 12. Widok panelu *Elements* z zaznaczonym elementem nagłówka GRANIT



Ryc. 13. Widok panelu *Styles* ze stylami dotyczącymi wybranego elementu nagłówka

3.2. Edycja "na żywo" (*Live Editing*)

DevTools pozwala na bezpieczne "psucie" strony bez zmieniania pliku źródłowego.

- 3.2.1. Znajdź w panelu *Styles* regułę dla `.item1` i kliknij na fioletowy kwadracik przy właściwości `background-color`. Wybierz inny kolor.

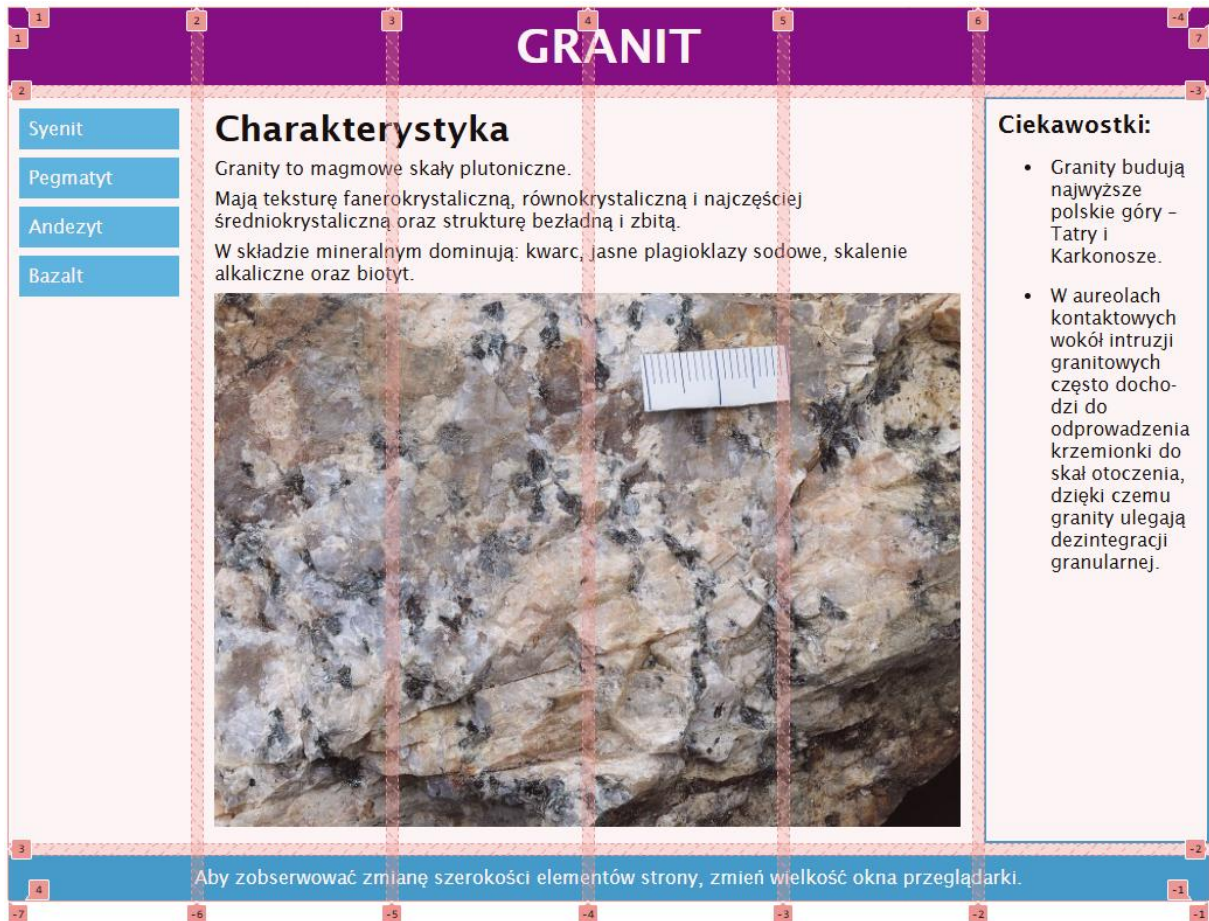
Live editing to najlepszy sposób na szybkie testowanie kolorów i marginesów przed wpisaniem ich do pliku `.css`.

3.3. Debugowanie Grid Layout

Chrome ma genialne wsparcie dla CSS Grid.

- 3.3.1. W panelu *Elements* znajdź `<div class="grid-container">`. Zobaczysz obok niego mały przycisk **grid**. Kliknij go.

Na stronie pojawią się linie pomocnicze siatki oraz ich numery (Ryc. 14). Włączmy teraz widoczność nazw obszarów. Aby to zrobić, gdy masz zaznaczony kontener siatki w zakładce *Elements*, musisz przejść do dedykowanego panelu ustawień *Grida*.

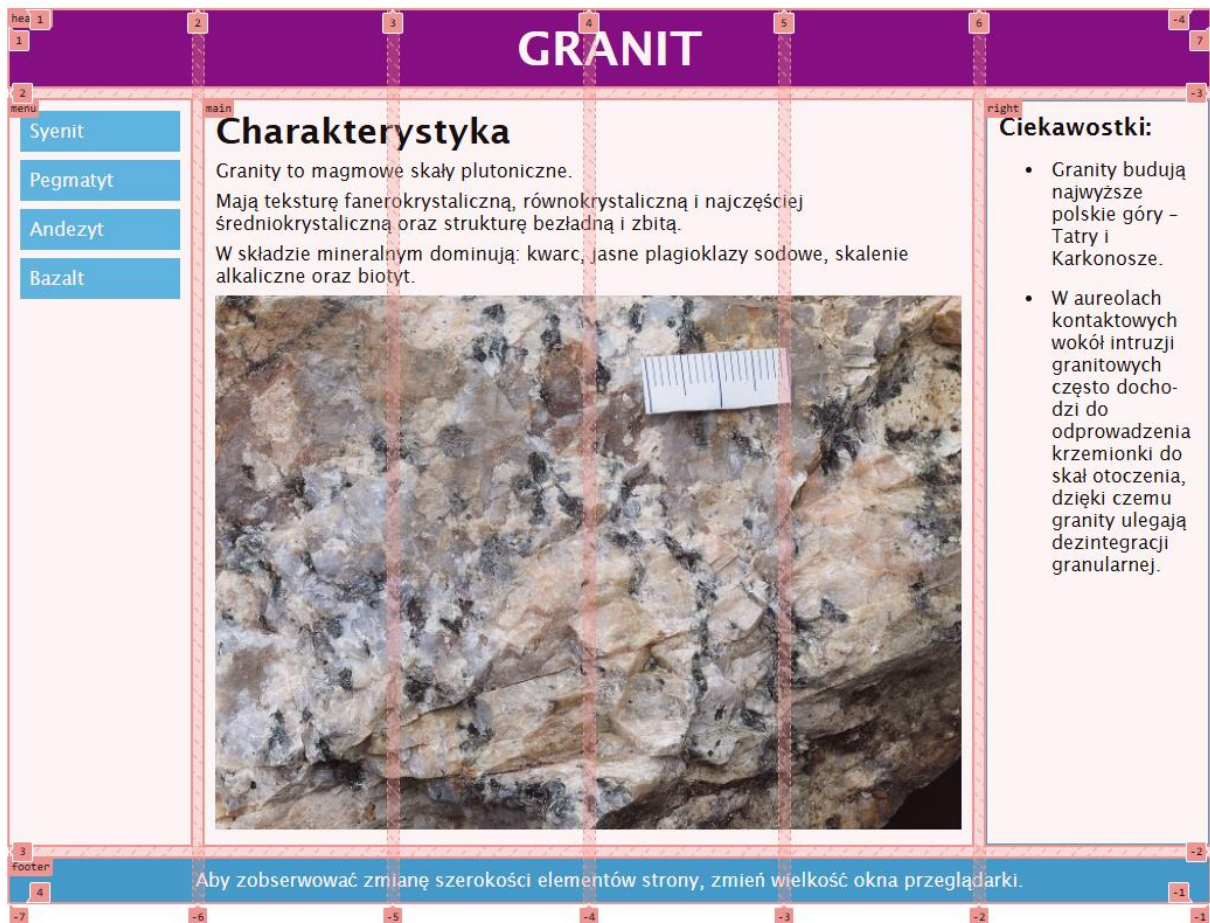


Ryc. 14. Widok projektu strony www z widocznym podziałem *grid*

3.3.2. W górnej części panelu *DevTools* (obok zakładek *Styles* i *Computed*) otwórz zakładkę *Layout*.

3.3.3. Odszukaj ustawienie *Overlay display settings* i zaznacz w nim opcję *Show area names*.

Teraz obok widoczne są także nazwy obszarów (*header*, *menu*, *main*) (Ryc. 15). Dzięki temu od razu widać, gdzie kończy się jedna kolumna, a zaczyna druga.



Ryc. 15. Widok projektu z podziałem grid, numerami linii oraz nazwami obszarów

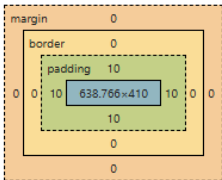
3.4. Zakładka *Computed* – Ostateczna prawda

Często studenci pytają: *"Dlaczego mój element ma taką szerokość, skoro wpisałem inną?"*. Odpowiedź na to pytanie odnajdziemy w zakładce *Computed*.

3.4.1. Kliknij na elemencie `item3` i w panelu bocznym wybierz zakładkę ***Computed***.

Zobaczysz tam model pudełkowy wybranego elementu z dokładnie wyliczonymi marginesami, paddingami i obramowaniami. Poniżej znajduje się lista wszystkich stylów (także tych dziedziczonych) bloku `item3` (Ryc. 16).

Styles **Computed** Layout Event Listeners DOM Breakpoints Properties Accessibility



margin 0
border 0
padding 10
0 10 638.766px 410 10 0 0
10
0

Filter Show all Group

- ▶ **box-sizing** border-box
- ▶ **display** block
- ▶ **font-family** "Lucida Sans", sans-serif
- ▶ **font-size** 16px
- ▶ **grid-column-end** main
- ▶ **grid-column-start** main
- ▶ **grid-row-end** main
- ▶ **grid-row-start** main
- height** 430px
- ▶ **margin-bottom** 0px
- ▶ **margin-left** 0px
- ▶ **margin-right** 0px
- ▶ **margin-top** 0px
- ▶ **padding-bottom** 10px
- ▶ **padding-left** 10px
- ▶ **padding-right** 10px
- ▶ **padding-top** 10px
- ▶ **unicode-bidi** isolate
- width** 658.766px

Rendered Fonts

Family name: Lucida Sans
PostScript name: LucidaSans
Font origin: Local file (235 glyphs)

Family name: Arial
PostScript name: ArialMT
Font origin: Local file (15 glyphs)

Family name: Lucida Sans
PostScript name: LucidaSans-Demi
Font origin: Local file (15 glyphs)

Ryc. 16. Zakładka *Computed* wyświetlająca właściwości bloku

Sprawdź w kodzie po jakim elemencie blok `item3` dziedziczy właściwości `padding`?

3.5. Symulacja wolnego internetu (*Network Throttling*)

Responsywność to nie tylko wygląd, to też wydajność na telefonie w lesie.

3.5.1. Przejdź do zakładki **Network**, znajdź pole "No throttling" i zmień je na **Fast 3G** lub **Slow 3G** i następnie odśwież stronę.

Sprawdź szybkość ładowania strony www, pliku styli oraz grafiki. Studenci zobaczą, jak długo ładuje się ich obrazek `granitoid.jpg` na słabym łączu mobilnym.

3.6. Symulacja zachowania strony przy wyświetlaniu na urządzeniach mobilnych

3.6.1. Kliknij ikonę **Toggle Device Toolbar** (na prawo od ikony *Select*).

3.6.2. Chwyć za prawą krawędź okna podglądu i powoli zwiężaj stronę.

- Obserwuj licznik szerokości *Dimension Responsive* (w pikselach) na górze ekranu.

- Zauważ moment, w którym napisy w menu (`item2`) przestają się mieścić, a tekst w sekcji głównej staje się zbyt wąski.
- **Ten moment to Twój przyszły Breakpoint.**
- Zmień predefiniowane ustawienie szerokości ekranu *Dimension Responsive* na np. **iPhone 12 Pro** i inne urządzenia.

Przy zmianie szerokości viewport poszczególne bloki ulegają stopniowej kompresji. Nie wygląda to dobrze. Najlepiej byłoby gdyby bloki w urządzeniach o małej szerokości ekranu inaczej układały bloki na stronie. W kolejnym etapie ćwiczenia postaramy się wskazać przeglądarce jak ma wyświetlać kolejne elementy strony w zależności od szerokości ekranu urządzenia (breakpoints).

4. Media Queries

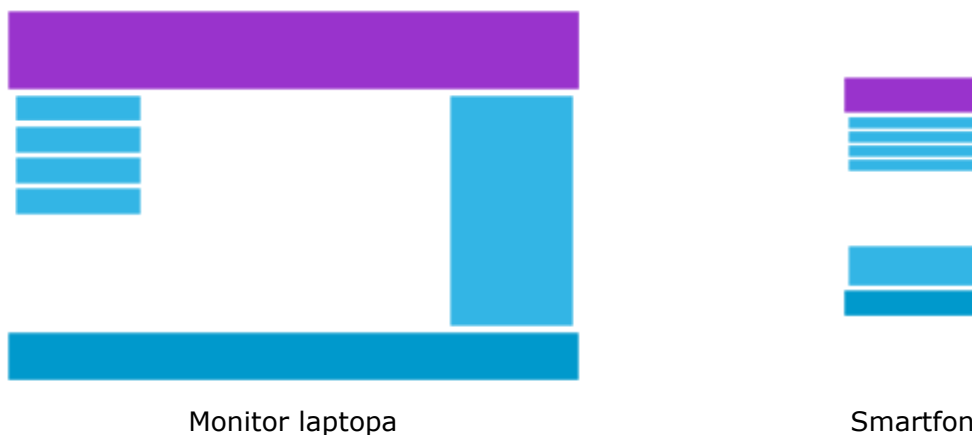
Media query to technika CSS wprowadzona w CSS3. Używa reguły `@media` aby uwzględnić blok właściwości CSS tylko wtedy, gdy spełniony jest pewien warunek.

Jeśli okno przeglądarki ma `600px` lub mniej, kolor tła będzie jasnoniebieski:

```
@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

4.1. Dodawanie punkt przerwania

Breakpoint (punkt przerwania) to fragment kodu CSS, który modyfikuje wygląd strony w zależności od szerokości ekranu (lub innych parametrów urządzenia) (Ryc. 17). Używa się do tego **media queries**, czyli zapytań medialnych. Dzięki nim strona może być **responsywna** — czyli dobrze wyglądać na różnych urządzeniach (telefony, tablety, komputery).



Ryc. 17. Różne ułożenie bloków modelu pudełkowego w zależności od szerokości obsługiwanej urządzenia

Wcześniej utworzyliśmy stronę internetową z wierszami i kolumnami siatki grid, która była responsywna, ale na skutek kompresji bloków nie wyglądała dobrze na małym ekranie. **Media queries** mogą w tym pomóc. Możemy dodać **Breakpoint**, w którym pewne części projektu będą zachowywać się inaczej po każdej stronie punktu przerwania. Chodzi o różne ułożenie poszczególnych bloków siatki w zależności od aktualnej szerokości ekranu.

4.1.1. Dodaj poniższy kod na koniec pliku CSS.

```
/* 1. DOMYŚLNIE (Smartfon - najmniejszy ekran) */
@media only screen and (max-width: 600px) {
  .item1 {grid-area: 1 / span 6;}
  .item2 {grid-area: 2 / span 6;}
  .item3 {grid-area: 3 / span 6;}
  .item4 {grid-area: 4 / span 6;}
  .item5 {grid-area: 5 / span 6;}
}
```

Powyższy kod to zapytanie medialne dodające punkt przerwania dla urządzeń gdy ich szerokość ekranu jest mniejsza niż 600px. Odpowiada to smartfonom. Jak rozumieć powyższe zapisy?...

@media only screen and (max-width: 600px) mówi: "zastosuj poniższe style tylko wtedy, gdy szerokość ekranu wynosi **600px lub mniej**".

.itemX { grid-area: ... } Każdy z elementów klas .item1, .item2, itd. dostaje nową pozycję w siatce CSS Grid.

grid-area: 1 / span 6;

to skrócony zapis właściwości CSS Grid. Składnia: grid-area: row-start / column-start / row-end / column-end, ale jeśli podasz tylko dwie wartości, tak jak tutaj, to: grid-area: <row-start> / <column-start>.

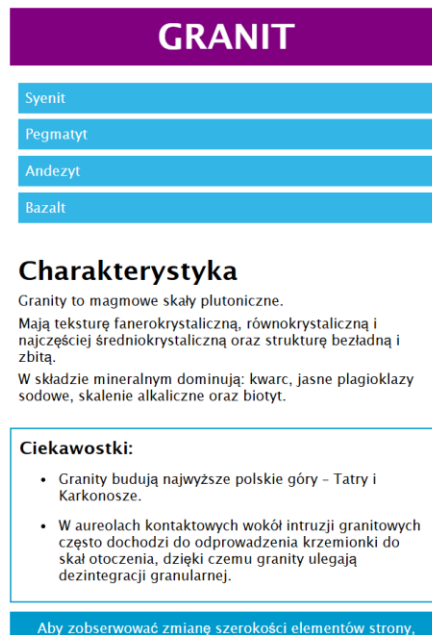
1 / span 6 oznacza:

- o zacznij w **1. wierszu**
- o zacznij w **kolumnie, która ma zająć 6 kolumn (span 6)**

Więc każdy .itemX układany jest jeden pod drugim (bo mają różne row-start), a wszystkie rozciągają się na 6 kolumn (czyli pełną szerokość w responsywnym widoku).

4.1.2. Przeładuj stronę w przeglądarce internetowej.

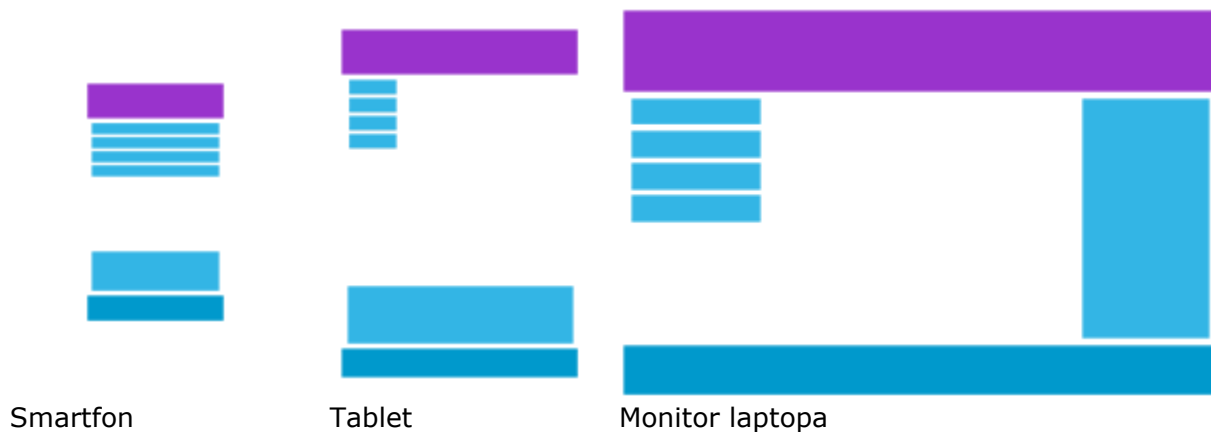
4.1.3. W DevTools przetestuj zachowanie strony internetowej przy zmianie szerokości ekranu. Zwróć szczególną uwagę na zachowanie strony poniżej i powyżej breakpointa 600px (**Ryc. 18**).



Ryc. 18. Widok responsywnego układu strony o szerokości max 600px

4.2. Kolejny Breakpoint

Możesz dodać tyle punktów przerwania ile chcesz. My w naszym ćwiczeniu dodamy jeszcze punkt przerwania dla tabletów (**Ryc. 19**).



Ryc. 19. Projekty strony dla trzech różnych szerokości ekranów

Wykorzystamy następujące punkty przerwania:

Smartfon	maks. 600px
Tablet	min 600px
Monitor	min 768px

UWAGA! Ważna zasada: Porządek w Media Queries

CSS (*Cascading Style Sheets*) to arkusze **kaskadowe**, co oznacza, że reguły czytane są od góry do dołu. Jeśli dwie reguły pasują do tej samej szerokości ekranu, wygrywa ta, która jest w pliku niżej.

Aby uniknąć chaosu, developerzy stosują zasadę **Mobile First**. Chodzi w niej o to aby na początku zdefiniować style dla najmniejszych ekranów (**bez użycia @media**), a następnie dodawać zapytania @media (min-width: ...) od najmniejszej wartości do największej.

Dzięki temu większe ekrany "dziedziczą" wspólne style z mniejszych i nadpisują tylko to, co faktycznie musi się zmienić (np. liczbę kolumn w siatce). Jeśli pomieszasz kolejność (np. dasz min-width: 1000px przed min-width: 600px), to styl dla 600px może nadpisać ten dla 1000px, psując wygląd strony na dużych monitorach!

4.2.1. Podmień kod CSS z punktu 4.1.1. na następujący:

```
/* 1. DOMYŚLNIE (Smartfon - najmniejszy ekran) */
.item1 {grid-area: 1 / span 6;}
.item2 {grid-area: 2 / span 6;}
.item3 {grid-area: 3 / span 6;}
.item4 {grid-area: 4 / span 6;}
.item5 {grid-area: 5 / span 6;}

/* 2. TABLET (Ekran powyżej 600px) */
@media only screen and (min-width: 600px) {
.item1 {grid-area: 1 / span 6;}
.item2 {grid-area: 2 / span 1;}
.item3 {grid-area: 2 / span 4;}
.item4 {grid-area: 3 / span 6;}
.item5 {grid-area: 4 / span 6;}
}

/* 3. MONITOR (Ekran powyżej 768px) */
@media only screen and (min-width: 768px) {
.item1 {grid-area: 1 / span 6;}
.item2 {grid-area: 2 / span 1;}
.item3 {grid-area: 2 / span 4;}
.item4 {grid-area: 2 / span 1;}
.item5 {grid-area: 3 / span 6;}
}
```

Gdy zostanie rozpoznany tablet wtedy:

Blok HTML	Ułoż w wierszu	Liczba kolumn do zajęcia
.item1	1	6
.item2	2	1
.item3	2	4

<code>.item4</code>	3	6
<code>.item5</code>	4	6

Gdy zostanie rozpoznany monitor wtedy:

Blok HTML	Ułoż w wierszu	Liczba kolumn do zajęcia
<code>.item1</code>	1	6
<code>.item2</code>	2	1
<code>.item3</code>	2	4
<code>.item4</code>	2	1
<code>.item5</code>	4	6

4.2.2. Przeładuj stronę w przeglądarce internetowej.

4.2.3. W DevTools przetestuj zachowanie strony internetowej przy zmianie szerokości ekranu. Zwróć szczególną uwagę na zachowanie strony poniżej i powyżej breakpointa 600px oraz poniżej i powyżej 768px.

4.3. Typowe punkty przerwania urządzenia

Istnieje mnóstwo ekranów i urządzeń o różnych wysokościach i szerokościach więc trudno jest utworzyć dokładny punkt przerwania dla każdego urządzenia. Aby uprościć sprawę, możesz na swojej stronie www wybrać pięć grup:

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

5. Interaktywność w RWD

Nasza strona www działa już całkiem nieźle. Na koniec sprawdzimy, że menu będzie zachowywać się jak w nowoczesnych aplikacjach mobilnych. Zamiast zajmować miejsce na małym ekranie, zostanie ukryte pod przyciskiem i rozwinię się dopiero po kliknięciu. Wykorzystamy do tego technikę *toggle* oraz prosty skrypt JavaScript.

5.1. Dodanie interaktywnego menu

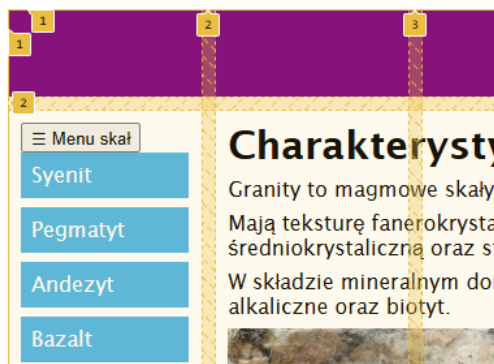
Obecnie menu (lista skał w `.item2`) jest zawsze widoczne. Na bardzo małych ekranach smartfonów zajmuje ono cenne miejsce w pionie. Naszym zadaniem będzie je ukryć i pokazać tylko wtedy, gdy użytkownik tego zechce.

Najpierw do naszego projektu dodamy przycisk. Wykorzystamy do tego symbol ☰ (encja: `☰`), ze względu na swój kształt powszechnie nazywany "hamburgerem".

5.1.1. W pliku HTML, wewnątrz sekcji `.item1` dodaj przycisk, który będzie służył do otwierania menu i nadaj liście `ul` identyfikator `.menu-list`.

```
<div class="item2">
  <button class="hamburger" onclick="toggleMenu()">☰ Menu skał</button>
  <ul id="menu-list">
    <li>Syenit</li>
    <li>Pegmatyt</li>
    <li>Andezyt</li>
    <li>Bazalt</li>
  </ul>
</div>
```

Po przeładowaniu strony www w oknie podglądu pojawił się przycisk z hamburgerem (Ryc. 20).



Ryc. 20. Dodanie przycisku ponad menu strony

Na dużych ekranach (monitorach) przycisk jest niepotrzebny. Powinien pojawiać się tylko przy niższych szerokościach viewpoint. Dlatego w CSS domyślnie ukryjemy go.

5.1.2. Dodaj do głównej sekcji CSS stylizację przycisku:

```
.hamburger {
  display: none; /* Ukrywamy przycisk na dużych ekranach */
  width: 100%;
  padding: 10px;
  background-color: #33b5e5;
  color: white;
  border: none;
```

```
cursor: pointer;
font-size: 18px;
}
```

5.1.3. Odśwież przeglądarkę. Przycisk powinien zniknąć.

Teraz zmodyfikujemy domyślną sekcję definiującą sposób wyświetlania bloków na ekranach smartfonów. Chcemy, aby przycisk się pojawił, a lista domyślnie zniknęła.

5.1.4. Do sekcji „1. DOMYŚLNIE (Smartfon - najmniejszy ekran)” dopisz:

```
/* 1. DOMYŚLNIE (Smartfon - najmniejszy ekran) */
.item1 {grid-area: 1 / span 6;}
.item2 {grid-area: 2 / span 6;}
.item3 {grid-area: 3 / span 6;}
.item4 {grid-area: 4 / span 6;}
.item5 {grid-area: 5 / span 6;}
.hamburger {
  display: block; /* Pokazujemy przycisk na telefonie */
}
#menu-list {
  display: none; /* Ukrywamy listę skał na starcie */
}
/* Klasa pomocnicza, którą dodamy za pomocą JavaScript */
#menu-list.show {
  display: block; /* Pokaż, gdy JS doda klasę .show */
}
```

Zauważ, że ta sekcja nie ma nawiasów, wobec czego odwracamy wcześniejszy ukrycie przycisku (`display: none;`). Każemu przyciskowi pojawić się (`display: block;`), ukrywamy identyfikator `#menu-list` (`display: none;`) oraz dodajemy pomocniczą klasę

Zgodnie z zasadą *mobile first* oraz rosnącym porządkiem szerokości ekranu, zajmijmy się teraz zachowaniem menu dla tabletów.

5.1.5. Ponownie ukryjmy przycisk i przywróćmy widoczność menu. Dodaj do sekcji smartfonów deklarację:

```
/* 2. TABLET (Ekran powyżej 600px) */
@media only screen and (min-width: 600px) {
  .item1 {grid-area: 1 / span 6;}
  .item2 {grid-area: 2 / span 1;}
  .item3 {grid-area: 2 / span 4;}
  .item4 {grid-area: 3 / span 6;}
  .item5 {grid-area: 4 / span 6;}
}
/* PRZYWRACANIE MENU DLA DUŻYCH EKRAŃÓW: */
.hamburger {
  display: none; /* Na tablecie przycisk jest już zbędny */
}
#menu-list {
```

```
display: block; /* Przywracamy widoczność listy na stałe */
}
}
```

Dla urządzeń o szerokości ekranu większych od 768px nie są potrzebne żadne deklaracje. Właściwości przycisku oraz menu zostaną odziedziczone od deklaracji dla tabletów.

Na koniec aby przycisk działał, musimy dopisać prostą funkcję, która będzie "przełączać" widoczność listy.

5.1.6. Dodaj ten kod na samym dole pliku HTML, przed zamknięciem znacznika `</body>`:

```
<script>
function toggleMenu() {
  var x = document.getElementById("menu-list");
  if (x.className === "") {
    x.className = "show";
  } else {
    x.className = "";
  }
}
</script>
```

Funkcja `toggleMenu` działa jak włącznik światła: jeśli lista nie ma klasy `show`, to ją dodaje (pokazuje menu), a jeśli już ją ma – zabiera ją (ukrywa menu).

5.1.7. Nasz projekt jest gotowy. Przetestuj zachowanie strony www dla różnych szerokości ekranu.

6. DODATKI

6.1. Orientacja: Pionowa/Pozioma

Media queries mogą być również używane do zmiany układu strony w zależności od orientacji przeglądarki. Możesz mieć zestaw właściwości CSS, które będą miały zastosowanie tylko wtedy, gdy okno przeglądarki jest szersze niż jego wysokość, tak zwana orientacja „Pozioma”:

Przykład

```
@media only screen and (orientation: landscape) {
  body {
    background-color: lightblue;
  }
}
```

Strona internetowa będzie miała jasnoniebieskie tło, jeśli orientacja będzie w trybie poziomym.

6.2. Ukrywanie elementów za pomocą media queries

Innym powszechnym zastosowaniem media queries jest ukrywanie elementów na ekranach o różnych rozmiarach:

```
/* If the screen size is 600px wide or less, hide the element */
@media only screen and (max-width: 600px) {
  div.example {
    display: none;
  }
}
```

6.3. Zmiana rozmiaru czcionki za pomocą media queries

```
/* If the screen size is 601px or more, set the font-size of <div> to 80px
*/
@media only screen and (min-width: 601px) {
  div.example {
    font-size: 80px;
  }
}

/* If the screen size is 600px or less, set the font-size of <div> to 30px
*/
@media only screen and (max-width: 600px) {
  div.example {
    font-size: 30px;
  }
}
```

6.4. Korzystanie z właściwości width

Obrazy mogą być płynnie skalowane w zależności od szerokości okna przeglądarki.

Jeśli właściwość `width` jest ustawiona na wartość procentową, a właściwość `height` jest ustawiona na „auto”, obraz będzie responsywny i skalowany w górę i w dół:

```
img {
  width: 100%;
  height: auto;
}
```

Zauważ, że w powyższym przykładzie obraz może zostać powiększony do rozmiaru większego niż oryginalny. Lepszym rozwiązaniem w wielu przypadkach będzie użycie właściwości `max-width`.

6.5. Korzystanie z właściwości `max-width`

Jeśli właściwość `max-width` jest ustawiona na `100%`, obraz zostanie zmniejszony, jeśli będzie to konieczne, ale nigdy nie zostanie powiększony do rozmiaru większego niż oryginalny:

```
img {
  max-width: 100%;
  height: auto;
}
```

6.5.1. Dodaj skalowanie obrazu do przykładowej strony internetowej

```
img {
  max-width: 100%;
  height: auto;
}
```

6.6. Obrazy tła

Obrazy tła mogą również reagować na zmianę rozmiaru i skalowanie.

Oto trzy różne metody:

1. Jeśli właściwość `background-size` jest ustawiona na „`contain`”, obraz tła zostanie przeskalowany i spróbuje dopasować się do obszaru zawartości. Jednak obraz zachowa swój współczynnik proporcji (proporcjonalny związek między szerokością i wysokością obrazu):

```
div {
  width: 100%;
  height: 400px;
  background-image: url('img_flowers.jpg');
  background-repeat: no-repeat;
  background-size: contain;
  border: 1px solid red;
}
```

2. Jeśli właściwość `background-size` jest ustawiona na „`100% 100%`”, obraz tła zostanie rozciągnięty, aby pokryć cały obszar zawartości:

```
div {
  width: 100%;
  height: 400px;
  background-image: url('img_flowers.jpg');
  background-size: 100% 100%;
  border: 1px solid red;
}
```

3. Jeśli właściwość `background-size` jest ustawiona na „cover”, obraz tła zostanie przeskalowany, aby pokryć cały obszar zawartości. Należy zauważyć, że wartość „cover” zachowuje współczynnik proporcji, a część obrazu tła może zostać przycięta:

```
div {
  width: 100%;
  height: 400px;
  background-image: url('img_flowers.jpg');
  background-size: cover;
  border: 1px solid red;
}
```

6.7. Różne obrazy dla różnych urządzeń

Duży obraz może być idealny na dużym ekranie komputera, ale bezużyteczny na małym urządzeniu. Po co ładować duży obraz, skoro i tak trzeba go pomniejszyć? Aby zmniejszyć obciążenie lub z innych powodów, możesz użyć zapytań medialnych, aby wyświetlać różne obrazy na różnych urządzeniach.

Oto jeden duży obraz i jeden mniejszy obraz, które będą wyświetlane na różnych urządzeniach:

```
/* For width smaller than 400px: */
body{
  background-image: url('img_smallflower.jpg');
}

/* For width 400px and larger: */
@media only screen and (min-width: 400px){
  body{
    background-image: url('img_flowers.jpg');
  }
}
```

Możesz użyć zapytania medialnego `min-device-width`, zamiast `min-width`, które sprawdza szerokość urządzenia, a nie szerokość przeglądarki. Wtedy obraz nie zmieni się, gdy zmienisz rozmiar okna przeglądarki:

```
/* For devices smaller than 400px: */
body {
  background-image: url('img_smallflower.jpg');
}

/* For devices 400px and larger: */
@media only screen and (min-device-width: 400px) {
  body {
    background-image: url('img_flowers.jpg');
  }
}
```

6.8. Element HTML `<picture>`

Element HTML `<picture>` daje twórcom stron internetowych większą elastyczność w określaniu zasobów obrazów.

Najczęstszym zastosowaniem elementu `<picture>` będą obrazy używane w projektach responsywnych. Zamiast mieć jeden obraz skalowany w górę lub w dół w zależności od szerokości pola widzenia, można zaprojektować wiele obrazów, aby ładniej wypełnić pole widzenia przeglądarki.

Element `<picture>` działa podobnie do elementów `<video>` i `<audio>`. Konfigurujesz różne źródła, a pierwsze źródło, które pasuje do preferencji, jest używane:

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 400px)">
  <source srcset="img_flowers.jpg">
  
</picture>
```

Atrybut `srcset` jest wymagany i definiuje źródło obrazu.

Atrybut `media` jest opcjonalny i akceptuje zapytania medialne, które znajdziesz w regule CSS `@media`.

Powinieneś również zdefiniować element `` dla przeglądarek, które nie obsługują elementu `<picture>`.

7. Wykorzystane materiały

Hughes J., 2022. Generate Media Queries for Specific Devices with this Insanely Simple CSS Tool. URL: <https://www.elegantthemes.com/blog/resources/generate-media-queries-for-specific-devices-with-this-insanely-simple-css-tool>.

Kodzisz.pl, 2015. Podstawy responsywnych stron WWW. URL: <https://www.youtube.com/watch?v=6benQW541p0>.

Simple CSS Media Query Generator. URL: <https://simplecss.eu/>.

W3C. Responsive Web Design. URL: https://www.w3schools.com/css/css_rwd_grid.asp.

6.

Przykład z jednostkami rem zamiast px

W punkcie 3.6 zmieniasz rozmiar czcionki na 80px i 30px.

- **Sugestia:** Wspomnij, że w RWD dobrą praktyką jest używanie rem. Dzięki temu, gdy użytkownik zmieni rozmiar czcionki w ustawieniach systemu/przeglądarki (np. ze względu na wadę wzroku), cała strona przeskaluje się poprawnie.

8. Dodanie interaktywności (Flexbox)

Choć ćwiczenie skupia się na Gridzie, warto dodać jedno zdanie o **Flexboxie** jako uzupełnieniu Grida. Grid jest świetny do układu całej strony (layoutu), ale Flexbox jest lepszy do ułożenia elementów wewnątrz nagłówka (header) czy stopki (footer).