


Politechnika Krakowska im. Tadeusza Kościuszki Wydział Mechaniczny Instytut Informatyki Stosowanej M-07 Katedra Systemów Informatycznych i Modelowania Komputerowego		
Laboratorium: responsive design przy zastosowaniu HTML i CSS		
Proj i prog front-end	Nr. lab.: 02-03, 4 x 45 minut	dr hab inż. Grzegorz Filo dr inż. Paweł Lempa

1. Cel laboratorium

Celem ćwiczeń jest opanowanie umiejętności projektowania i programowania wyglądu responsywnych stron internetowych przy wykorzystaniu języka znaczników HTML5 i kaskadowych arkuszy styli CSS3 w środowisku VS Code.

2. Wiadomości wymagane do zaliczenia testu

Test będzie wymagał wpisania nazw poleceń / znaczników / atrybutów o podanym opisie z zakresu:

- podstawowe polecenia git, składnia, parametry i zastosowanie: config, init, status, add, commit, remote, pull, push,
- znaczniki HTML5: typ dokumentu, komentarz, główny element (!DOCTYPE, !--, html), metadane (head, title, base, link, meta, style), sekcje (body, article, section, nav, h1 ... h6, header, footer), grupowanie treści (p, hr, ol, ul, li, div), semantyka tekstu (a, em, strong, i, b, u, br, q, sub, sup), osadzanie obiektów (img, video, audio, canvas, svg, math), tabele (table, caption, col, tbody, tr, td, th), formularze (form, legend, label, input, button, select, option, datalist);
- atrybuty HTML5: alt, charset, class, content, controls, form, height, hidden, href, id, lang, method, name, rel, src, style, target, title, type, width
- atrybuty modelu CSS box: width, height, padding, border, margin
- selektory CSS: .class, *, #, :active, :after, :before, :hover, :link :root, :visited, element, [attribute]
- tło: background-color, background-image, background-position, background-size
- właściwości: position, font, text-, float, flex-, grid-.

3. Wymagane oprogramowanie i wprowadzenie

Zadania należy wykonać w środowisku programowania *Visual Studio Code*.

Do zdefiniowania struktury dokumentu HTML można wykorzystać uniwersalny znacznik obszaru `<div>` i/lub specyficzne znaczniki HTML5 służące do wyodrębnienia kilku typowych obszarów:

- `<header>` - nagłówek (logo),
- `<section>` - definicja sekcji zawierającej docelowo dwa elementy w układzie poziomym: pasek boczny i panel zasadniczej treści,
- `<nav>` - pasek nawigacyjny,
- `<article>` - zasadnicza treść, paragrafy tekstu, obrazy itp.,
- `<footer>` - stopka strony.

W ramach niniejszego laboratorium zostanie przećwiczone wykorzystanie obu tych możliwości.

Uzyskanie złożonego, wielokolumnowego układu wyglądu dokumentu HTML wymaga również wykorzystania arkuszy styli CSS. Nowoczesne strony muszą też spełniać warunki responsywności,

czyli dostosowania sposobu prezentacji treści do szerokości ekranu. Można to uzyskać za pomocą układów CSS (CSS layouts):

- CSS Float Layout – najprostsza technika polegająca na odpowiednim przypisaniu właściwości CSS o nazwach `float` i `clear`; pozwala na sztywno rozmieszczać obszary obok siebie, co może zmniejszać elastyczność takiego rozwiązania;
- CSS Flexbox Layout – technika oparta na definiowaniu kontenerów (zwykle przy zastosowaniu znaczników `<div>`), w których zawartość jest wyświetlana elastycznie (wartość `flex` właściwości `display`). Gwarantuje znaczną przewidywalność wyglądu elementów również w przypadku, gdy układ strony musi uwzględniać różne rozmiary ekranu i różne urządzenia wyświetlające. Wadą tego rozwiązania jest jego jednowymiarowość, co nie pozwala na zupełnie dowolne kształtowanie obszarów dokumentu;
- CSS Grid Layout – obecnie najbardziej zaawansowana technika definiowania układu za pomocą arkuszy stylów CSS. Technika ta jest oparta na układzie w pełni 2-wymiarowym, z wierszami i kolumnami. Znacząco ułatwia projektowanie wyglądu stron internetowych bez konieczności używania elementów zmiennoprzecinkowych i pozycjonowania.

4. Zadania do wykonania

Należy wykonać zadania opisane w punktach 4.1 – 4.4. Efektem końcowym powinny być trzy działające prototypy strony internetowej o układzie pokazanym na poniższym diagramie, zbudowane kolejno zgodnie z trzema typami układu (CSS layout): Float (4.2), Flexbox (4.3) oraz Grid (4.4). Bazą będzie zbudowany szkielet strony przy wykorzystaniu HTML. Układ głównych obszarów powinien być następujący: nagłówek, poniżej układ wielokolumnowy, zawierający pasek boczny oraz zasadniczą treść wypełniającą resztę miejsca. Na dole stopka wypełniająca całą dostępną szerokość.

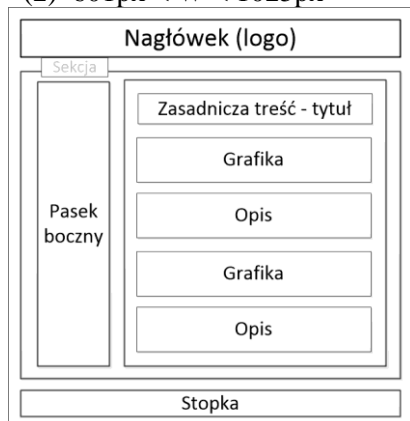
Docelowo układ zawartości powinien dostosowywać się automatycznie do szerokości ekranu:

- do 600px – cała zawartość w układzie pionowym,
- do 1024px – zasadnicza część dokumentu dwukolumnowa,
- powyżej 1024px – zasadnicza część dokumentu trójkolumnowa.

(1) $W < 601\text{px}$



(2) $601\text{px} < W < 1025\text{px}$



(3) $W > 1025\text{px}$



4.1 Budowa bazowej struktury w HTML

Należy zamknąć wszystkie otwarte katalogi, następnie utworzyć i otworzyć nowy katalog o nazwie PPF_Lab2. Następnie utworzyć podkatalogi `img` i `style` oraz pliki: w katalogu głównym `readme.md` i `index.html`, w podkatalogu `style` `style.css`, natomiast do podkatalogu `img` wgrać pliki multimedialne pobrane i wypakowane z archiwum znajdującego się w systemie elf (materiały do laboratorium 2, plik `img.zip`). {Obrazy zostały pobrane z darmowej galerii www.freeimages.com}. Należy zainicjować główną gałąź lokalnego repozytorium git i umieścić w niej pliki.

```

9 | <body>
10 |   <div>
11 |     <header>Header</header>
12 |
13 |     <section>
14 |       Section
15 |       <nav>Nav</nav>
16 |       <article>Article</article>
17 |     </section>
18 |
19 |     <footer>Footer</footer>
20 |   </div>
21 | </body>

```

Wprowadzone w tym momencie krótkie teksty powinny posłużyć do wizualnej oceny, czy strona wyświetla się poprawnie w przeglądarce.

Uwaga 1:

Kod na każdym etapie tworzenia strony powinien być odpowiednio formatowany, zgodnie z wybranym stylem. Zalecamy instalację rozszerzenia Prettier Code Formatter i zastosowanie go jako domyślnego narzędzia formatowania. Sformatowanie aktualnie aktywnego dokumentu przy wykorzystaniu aktualnego formatera: PPM (prawy przycisk myszy) → Format Document lub skrót klawiaturowy Shift+Alt+F. Natomiast polecenie Format Document With... pozwala na wybór lub zmianę domyślnego stylu formatowania kodu.

Uwaga 2:

W dowolnym momencie można obejrzeć aktualny wygląd strony np. otwierając za pomocą Live Server. Po uruchomieniu przeglądarki można jej nie zamykać, wykonanie każdego zapisu plików projektu spowoduje automatyczne odświeżenie zawartości.

Teraz nastąpi uzupełnienie treści strony przykładową zawartością:

- wewnątrz <head> zostanie dodany link do pliku CSS i zmieniony tytuł:

```

4 | <head>
5 |   <meta charset="UTF-8" />
6 |   <link rel="stylesheet" href="style/style.css" type="text/css" />
7 |   <title>PPF_Lab2</title>
8 | </head>

```

język dokumentu można zmienić na nieokreślony poprzez przypisanie atrybutu:

```
<html lang="zxx">
```

- znacznik <header> będzie zawierał logo zapisane w pliku img/logo.png o wymiarach 150px x 75px oraz tekst nagłówka <h1>: Projektowanie i programowanie front-end, lab. 2-3

```

12 |   <header>
13 |     <div>
14 |       
15 |     </div>
16 |     <h1>
17 |       Projektowanie i programowanie front-end, lab. 2-3
18 |     </h1>
19 |   </header>

```

- w kolejnym kroku nastąpi uzupełnienie zawartości znacznika `<section>`. Na początek należy usunąć tekst Section
- obszar `<nav>` zagnieżdżony wewnątrz `<section>` powinien zawierać listę kilku odsyłaczy różnych rodzajów:

```

25 |         <nav>
26 |             <ul>
27 |                 <li><a href="https://www.pk.edu.pl" target="_blank">Politechnika</a></li>
28 |                 <li><a href="https://www.agh.edu.pl" target="_blank">AGH</a></li>
29 |                 <li><a href="info.html" target="_self">Informacje</a></li>
30 |                 <li><a href="#animals">Zwierzęta</a></li>
31 |                 <li>
32 |                     <a href="https://www.youtube.com/watch?v=MhhAox6Zei8" target="_blank">Film</a>
33 |                 </li>
34 |                 <li><a href="#movie">Film lokalny</a></li>
35 |             </ul>
36 |         </nav>

```

- najbardziej rozbudowaną zawartość będzie miał znacznik `<article>`. Na początku zostanie umieszczony obszar `<div>` z tekstem nagłówka `<h2>`:

```

38 |         <article>
39 |             <div>
40 |                 <h2 id="animals">Kilka zwierząt żyjących na naszej planecie</h2>
41 |             </div>

```

następnie należy trzykrotnie powtórzyć sekwencję dwóch znaczników `<div>`, w których pierwszy zawiera obraz, a drugi – opis tekstowy. Poniżej podano przykład pierwszej sekwencji ze zdjęciem `img-01.png`. Należy **samodzielnie** dodać co najmniej kolejne dwa (w katalogu `img` powinno znajdować się pięć obrazów `img-01.png` do `img-05.png`) oraz wyszukiwać i wkleić analogiczne krótkie opisy tekstowe, na przykład:

```

42 |             <div>
43 |                 
44 |             </div>
45 |             <div>
46 |                 <p>
47 |                     Żaba wodna dorasta do 9 cm długości ciała. Tak samo jak w
48 |                     przypadku innych płazów bezogonowych, samiec jest mniejszy od
49 |                     samicy. Podczas godów samce przybierają lekko żółtawy odcień,
50 |                     podobnie jak żaba jeziorkowa.<br />
51 |                 </p>
52 |             </div>

```

- ostatnim obszarem zagnieżdżonym wewnątrz `<section>` jest `<div>` definiujący odtwarzacz pliku wideo (`mp4`). Plik `movie.mp4` jest zapisany w archiwum `img.zip`:

```

79 |             <div>
80 |                 <h2 id="movie">Krótki film o zwierzętach afrykańskich</h2>
81 |                 <video width="512" controls>
82 |                     <source src="img/movie.mp4" type="video/mp4" />
83 |                     Przeglądarka nie obsługuje video HTML.
84 |                 </video>
85 |             </div>
86 |         </section>

```

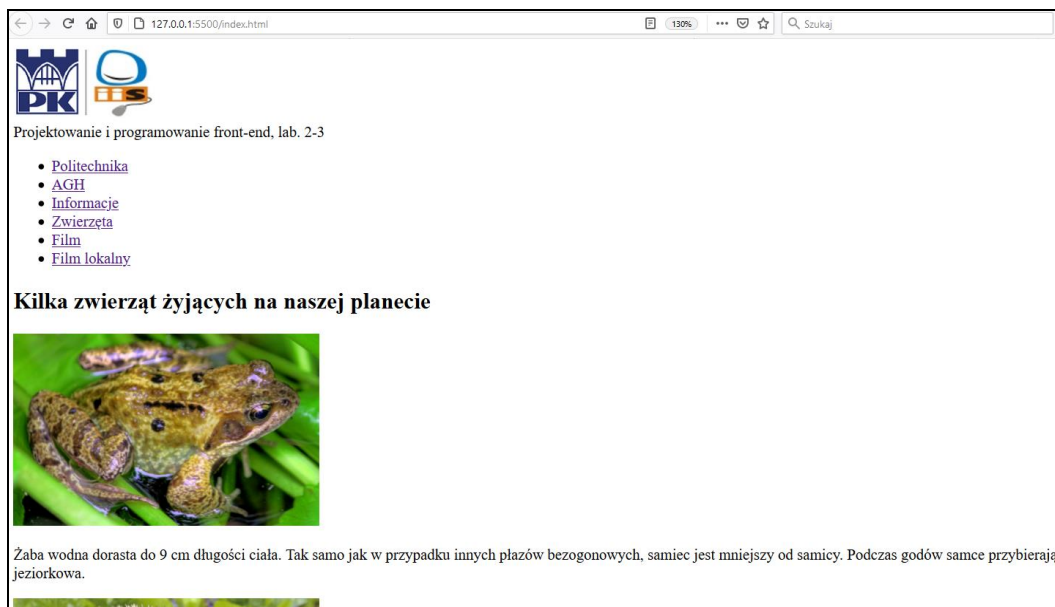
- za znacznikiem `<section>` znajduje się jeszcze obszar stopki zdefiniowany za pomocą znacznika `<footer>`:

```

88 | <footer>
89 |     <p>
90 |         Autor: <b>Imię Nazwisko</b>. Politechnika Krakowska, Wydział
91 |         Mechaniczny, Instytut Informatyki Stosowanej M-07
92 |     </p>
93 | </footer>

```

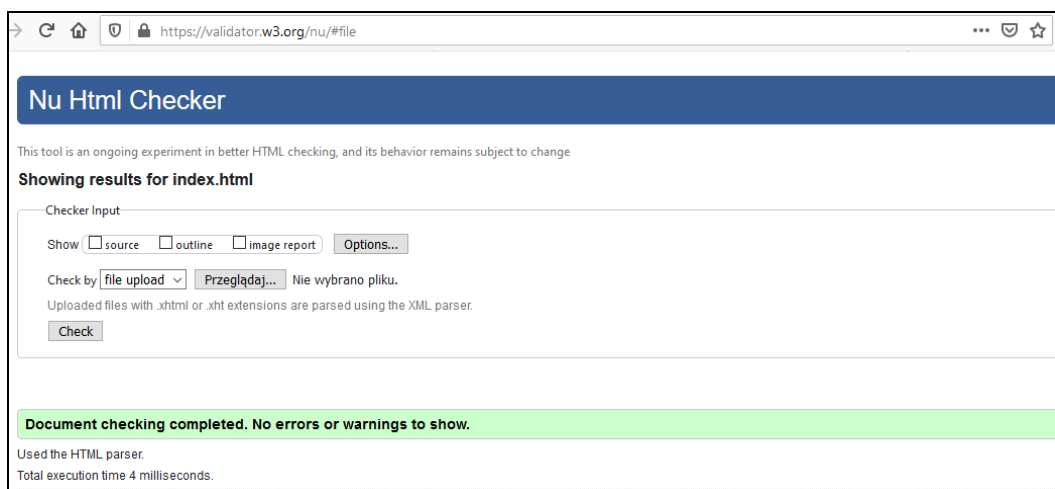
Fragment widoku strony w przeglądarce Firefox 81.0.2 bez zdefiniowanych stylów wyświetlania:

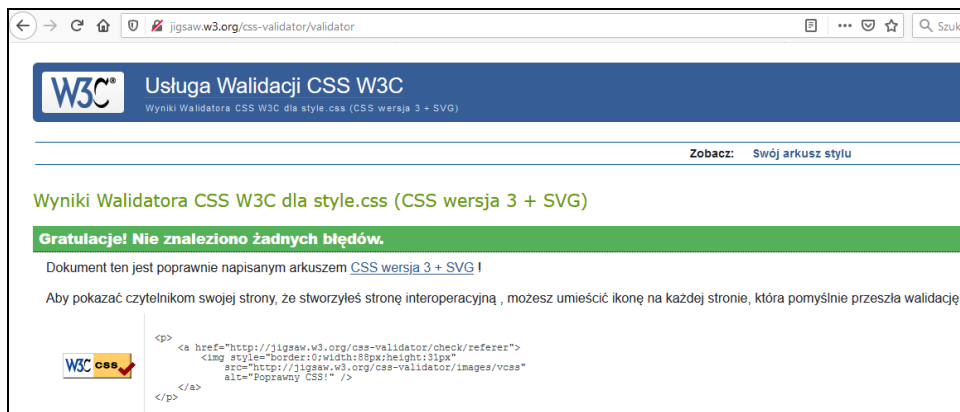


Należy jeszcze utworzyć plik o nazwie info.html w głównym katalogu projektu. W tym momencie może on zawierać dowolną poprawną treść, np. nagłówek `<h3>`, paragraf `<p>` zawierający krótki tekst (można wykorzystać *emmet abbreviation*) oraz link do głównego pliku html, np.: `POWRÓT`.

Wszystkie tworzone dokumenty HTML i CSS powinny pozytywnie przejść proces walidacji. Walidację dokumentów należy przeprowadzać odpowiednio w serwisach:

<https://validator.w3.org/> oraz <http://jigsaw.w3.org/css-validator/> :





4.2 Układ oparty na CSS Float Layout

Po pozytywnym przeprowadzeniu procesu walidacji należy zapisać utworzony dotychczas kod w repozytorium git w domyślnej gałęzi (master), a następnie utworzyć nową gałąź o nazwie PPF_LAB2_FLOAT. Należy upewnić się, że nowo utworzona gałąź jest aktywna (nazwa w lewym dolnym narożniku okna VSCode). Następnie należy kolejno wykonać niezbędne modyfikacje pliku HTML i CSS:

- w pliku index.html należy przypisać atrybuty class do znaczników <div> zagnieżdżonych wewnątrz <article>. Poniżej znajduje się przykład przypisania do pierwszego rysunku i opisu, analogicznie należy przypisać klasy article_image, article_image_src oraz article_text do pozostałych:

```

37 | <article>
38 |   <div id="animals">
39 |     <h2>Kilka zwierząt żyjących na naszej planecie</h2>
40 |   </div>
41 |   <div class="article_image">
42 |     
43 |   </div>
44 |   <div class="article_text">
45 |     <p>
46 |       Żaba wodna dorasta do 9 cm długości ciała. Tak samo jak w
47 |       przypadku innych płazów bezogonowych, samiec jest mniejszy od
48 |       samicy. Podczas godów samce przybierają lekko żółtawy odcień,
49 |       podobnie jak żaba jeziorkowa.<br />
50 |     </p>
51 |   </div>

```

Kolejne zmiany będą dotyczyły edycji zawartości pliku style.css. Uzyskanie wielokolumnowego wyglądu strony w oparciu o układ CSS Float Layout sprowadza się do przypisania odpowiednim znacznikom właściwości float o wartości left lub right. Skutkuje to ustawieniem obszarów obok siebie, zamiast domyślnego układu pionowego:

- na początek ustawienie zawarcia dopełnienia (padding) i obramowania (border) w ramach właściwości width i height, dotyczące wszystkich znaczników:

```

1 | * {
2 |   box-sizing: border-box;
3 | }

```

- dalej, ogólne ustawienia dotyczące czcionek i wyrównania tekstu, dotyczące znaczników <body>, <h2> i <p>:

```

5  body {
6  |   font-family: Arial, Helvetica, sans-serif;
7  | }
8  h2 {
9  |   font-size: 20px;
10 |   width: 100%;
11 | }
12 p {
13 |   text-align: justify;
14 | }

```

- obszarom nagłówka i stopki należy nadać następujące właściwości: wyświetlanie na całej dostępnej szerokości, szary kolor tła, wartość dopełnienia (padding) 10 pikseli, wyrównanie tekstu do środka, kolor czcionki ciemnoniebieski.

Najważniejsza z punktu widzenia układu właściwość: **float: left;** zostanie przypisana do obiektu obrazu `` zagnieżdżonego wewnątrz `<header>`. Poza tym należy przypisać stałą wielkość czcionki 28px dla nagłówka `<h1>` zagnieżdżonego wewnątrz bloku `<header>`.

```

16 header,
17 footer {
18 |   width: 100%;
19 |   background-color: #rgb(182, 182, 182);
20 |   padding: 10px;
21 |   text-align: center;
22 |   color: #rgb(45, 3, 230);
23 | }
24 header img {
25 |   float: left;
26 | }
27 header h1 {
28 |   font-size: 28px;
29 | }

```

- Po zmianie kierunku przepływu bloków dokumentu za pomocą właściwości `float`, należy pamiętać o przywróceniu domyślnej wartości. Praktycznie realizuje się to stosując pseudo-element `:after{}`:

```

32 header:after{
33 |   content: "";
34 |   display: block;
35 |   clear: both;
36 | }

```

- dla znacznika `section` należy ustawić tylko kolor tła na wartość `#cccccc`, a następnie dodać pseudo-element ustawiający domyślny przepływ, analogicznie do `header`,
- lista odnośników w bloku `<nav>` jest zbudowana w postaci wypunktowania. Blok ten powinien być wyświetlany po lewej stronie obok kolejnego, którym będzie `<article>`, więc należy użyć właściwości `float`. Szerokość należy ustawić na 20% dostępnego obszaru, kolor tła `#cccccc`, dopełnienie 20px. Usunięcie punktów można uzyskać przypisując właściwość `list-style-type`. Poszczególne elementy listy zostaną rozsunięte za pomocą wartości przypisanej do właściwości `padding-bottom`.

```

47 nav {
48 |   background-color: #cccccc;
49 |   float: left;
50 |   width: 20%;
51 |   padding: 20px;
52 | }

```

```

53  nav ul {
54      list-style-type: none;
55      padding: 0;
56  }
57  nav li {
58      padding-bottom: 12px;
59  }

```

- Ostatni zbiór właściwości dotyczy najbardziej rozbudowanego bloku – `article`. Blok ten należy umieścić na prawo od `nav`, stąd ponowne przypisanie właściwości `float: left`, w taki sposób, aby wypełniał całą pozostałą przestrzeń, więc jego szerokość powinna mieć wartość 80%.

```

61  article {
62      float: left;
63      padding: 20px;
64      width: 80%;
65      background-color: #f1f1f1;
66  }

```

- Niektóre znaczniki zagnieżdżone wewnątrz `article` zostały przypisane do klas, w celu łatwej identyfikacji i szybkiej zmiany ich właściwości. Klasa `article_image` dotyczy obszaru, w którym jest umieszczany obraz, `article_text` - obszaru z opisem tekstowym, natomiast `article_image_src` jest klasą dla znacznika `img`. Właściwości definiowane w poszczególnych klasach podano poniżej.
Właściwości zdefiniowane w tych klasach zapewniają, że grafiki są wyświetlane z lewej strony dostępnego obszaru i są opływane od prawej przez tekst. Poprawne wyświetlanie kolejnych obrazów wymaga resetu kierunku przepływu po każdym bloku tekstu. Należy **samodzielnie** dodefiniować pseudo-element `div.article_text:after{}`.

```

67  div.article_image {
68      float: left;
69      padding: 5px;
70  }
71  div.article_text {
72      padding: 20px;
73      width: 100%;
74      text-align: justify;
75  }
76  img.article_image_src {
77      width: 100%;
78  }

```

Po uzyskaniu satysfakcjonującego wyglądu strony należy sprawdzić, jak formatowana jest zawartość w przypadku zmniejszenia szerokości okna przeglądarki.

Podsumowanie sekcji 4.2

Użycie układu CSS Float Layout pozwala na pozycjonowanie elementów blokowych będących składowymi strony w kierunku poziomym (lewym lub prawym). Głównymi zaletami stosowania tego podejścia są:

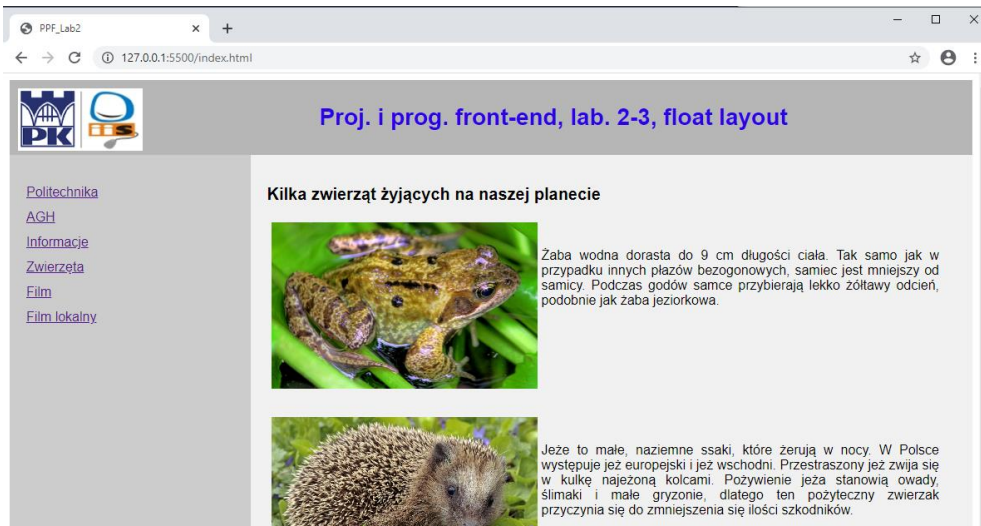
- bezproblemowa kompatybilność wsteczna, działa na wszystkich wersjach przeglądarek,
- łatwość stosowania,

Natomiast do wad zalicza się:

- konieczność stosowania dodatkowych elementów do resetowania kierunku przepływu,
- zmiana kierunku przepływu tylko dla niektórych elementów zaburza ogólny schemat przepływu całej strony,






- rozwiązanie to działa najlepiej przy opływaniu obrazów tekstem, ponieważ w tym celu zostało pierwotnie opracowane. Adaptacja do ogólnego stosowania z wszelkiego rodzaju elementami blokowymi została dodana później.

Fragment okna przeglądarki Chrome 86 z końcową wersją zadania 4.2

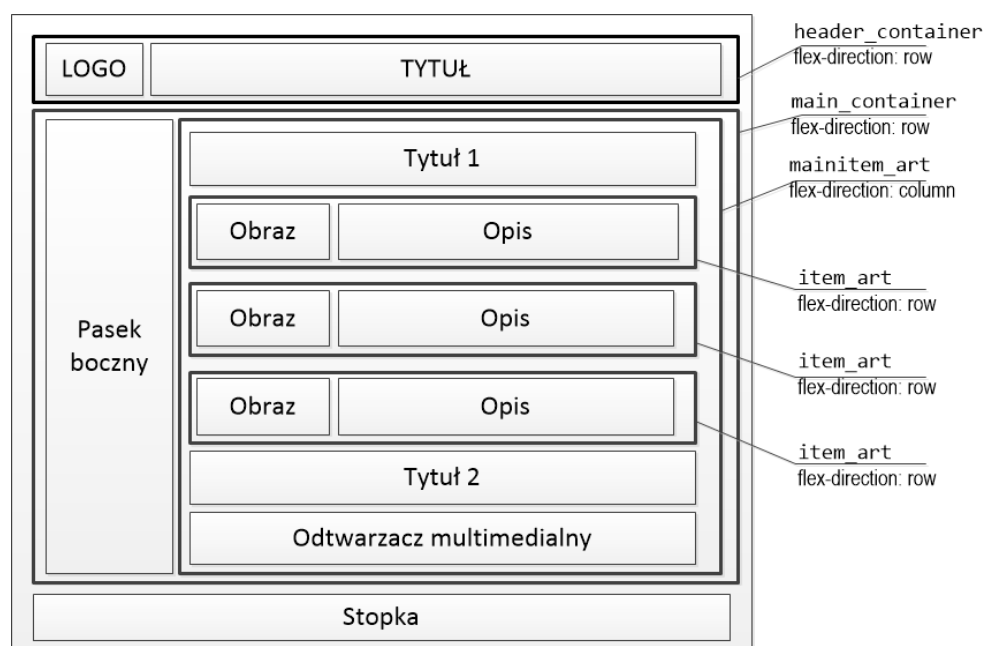


4.3 Układ oparty na CSS Flexbox layout

Układ wykorzystujący Flexbox jest obecnie wspierany przez wszystkie popularne przeglądarki. Poniżej przedstawiono minimalne wersje zapewniające wspieranie tej techniki:

				
29.0	11.0	22.0	10	48

Schemat układu kontenerów flexbox, nazwy klas i kierunki przepływu przedstawiono na rysunku:



Układ strony jest dosyć rozbudowany, stąd aby uzyskać zadowalający efekt wizualny zostanie kolejno zdefiniowanych 6 kontenerów flexbox. Będą to:

- kontener obszaru nagłówka `header_container` (układ poziomy): logo i tytuł strony,
- główny obszar zawartości `main_container` – pasek nawigacyjny (poprzednio `<nav>`) oraz obszaru z treścią (poprzednio `<article>`) (układ poziomy),
- obszar z treścią `mainitem_art` – kontener w układzie pionowym, w którym znajdują się: nagłówek artykułu, trzy obszary z obrazem i opisem, nagłówek filmu i odtwarzacz wideo,
- każdy obszar zawierający obraz i opis `item_art` jest kontenerem w układzie poziomym.

Zadanie należy rozpocząć od przejścia na gałąź `master`, a następnie utworzeniu nowej gałęzi `PPF_LAB2_FLEXBOX`. Struktura bazowego dokumentu `index.html` powinna zostać zmodyfikowana w taki sposób, aby uzyskać przedstawiony poniżej układ głównych znaczników.

```
9   <body>
10  <div>
11    <div class="header_container">
12      <div class="headeritem_logo">
13        
14      </div>
15      <div class="headeritem_text">
16        <h1>Proj. i prog. front-end, lab. flexbox</h1>
17      </div>
18    </div>
19    <div class="main_container">
20      <div class="mainitem_nav">...
37    </div>
38    <div class="mainitem_art">...
90  </div>
91  </div>
92  <div class="main_footer">...
97  </div>
98  </div>
99  </body>
```

Schemat zmian:

<code><header></code>	→	<code><div class="header_container"></code>
<code><div></code>	→	<code><div class="headeritem_logo"></code>
<code><h1>...</h1></code>	→	<code><div class="headeritem_text"><h1>...</h1></div></code>
<code><section></code>	→	<code><div class="main_container"></code>
<code><nav></code>	→	<code><div class="mainitem_nav"></code>
<code><article></code>	→	<code><div class="mainitem_art"></code>
<code><footer></code>	→	<code><div class="main_footer"></code>

Natomiast klasy trzech kontenerów `item_art` zawartych wewnątrz `mainitem_art` i ich elementów składowych należy zdefiniować wg przykładu:

```
42  <div class="item_art">
43    <div class="item_art_img">
44      
45    </div>
46    <div class="item_art_text">
47      <p>...
52    </p>
53  </div>
54  </div>
```

W pliku style.css należy zdefiniować kolejno:

- właściwości dotyczące podstawowych ustawień dopełnienia, obramowania, czcionek oraz akapitów tekstowych:

```
1  * {
2    | box-sizing: border-box;
3  }
4  body {
5    | font-family: Arial, Helvetica, sans-serif;
6    | padding: 5px;
7    | margin: 0;
8  }
9  h2 {
10   | font-size: 20px;
11 }
12 p {
13   | text-align: justify;
14   | font-size: 18px;
15 }
```

- następnie definiujemy przepływ w kolejnych kontenerach flexbox. Dla header_container:

```
17 .header_container {
18   | display: -webkit-flex;
19   | display: flex;
20   | flex-direction: row;
21   | flex-wrap: wrap;
22   | align-items: center;
23   | justify-content: center;
24   | align-content: center;
25   | background-color: #808080;
26   | padding: 10px;
27   | color: #000080;
28 }
29 .headeritem_logo {
30   | width: 180;
31 }
32 .headeritem_text {
33   | flex: 1 1 auto;
34 }
35 .headeritem_text h1 {
36   | font-size: 24px;
37   | text-align: center;
38 }
```

- dla main_container:

```
38 .main_container {
39   | display: -webkit-flex;
40   | display: flex;
41   | flex-direction: row;
42   | flex-wrap: nowrap;
43   | align-items: left;
44   | justify-content: left;
45   | align-content: left;
46 }
```

- dla mainitem_nav oraz obiektów zagnieżdżonych:

```

48 .mainitem_nav {
49     background-color: #cccccc;
50     flex: 1 1 auto;
51     align-self: stretch;
52     width: 20%;
53     min-width: 160px;
54     max-width: 320px;
55     padding: 20px;
56 }
57 .mainitem_nav ul {
58     list-style-type: none;
59     padding: 0;
60 }
61 .mainitem_nav li {
62     padding-bottom: 12px;
63 }

```

- dla `mainitem_art`:

```

65 .mainitem_art {
66     display: -webkit-flex;
67     display: flex;
68     flex-direction: column;
69     flex-wrap: nowrap;
70     align-items: stretch;
71     justify-content: stretch;
72     align-content: left;
73     padding: 20px;
74     background-color: #f1f1f1;
75 }

```

- dla `item_art` i elementów zagnieżdżonych:

```

77 .item_art {
78     display: -webkit-flex;
79     display: flex;
80     flex-direction: row;
81     flex-wrap: nowrap;
82     align-items: left;
83 }
84 .item_art_img {
85     flex: 1 1 auto;
86     width: 360px;
87     min-width: 360px;
88 }
89 .item_art_text {
90     align-self: stretch;
91 }

```

- oraz dla klasy stopki `main_footer`:

```

93 .main_footer {
94     width: 100%;
95     background-color: rgb(182, 182, 182);
96     padding: 10px;
97     text-align: center;
98     color: rgb(45, 3, 230);
99 }

```

W tym momencie można sprawdzić, jak zawartość strony jest rozmieszczana podczas zwiększania i zmniejszania szerokości okna przeglądarki. Ustawienie parametru `flex-wrap: nowrap;` powoduje zablokowanie możliwości przeniesienia obszaru do kolejnego wiersza/kolumny.

Uzyskanie responsywności zostanie w tym przypadku uzyskane poprzez zastosowanie *media queries*, które w wersji CSS3 pozwalają na używanie wyrażeń logicznych (*media features*). Wyrażenia te mogą zostać wykorzystane do sprawdzenia m.in. aktualnych wymiarów dostępnego obszaru i odpowiedniego dopasowanie układu wyświetlanej treści. W ramach laboratorium zostaną utworzone dwa wyrażenia:

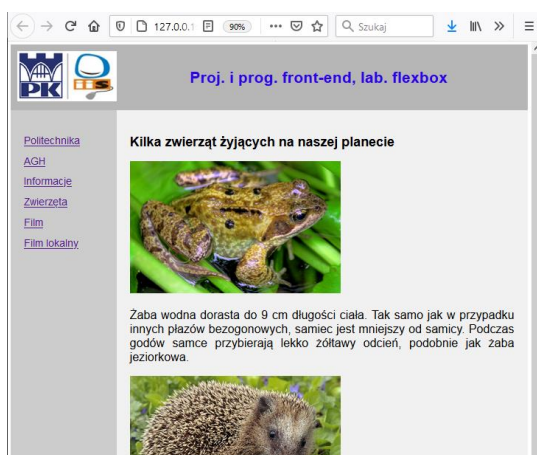
- jeżeli dostępna szerokość zmniejszy się poniżej 1025px, przepływ w kontenerze `item_art` zmieni się na pionowy (opisy znajdują się pod obrazami),

```
101 @media screen and (max-width: 1025px) {
102   .item_art {
103     -webkit-flex-direction: column;
104     flex-direction: column;
105   }
106 }
```

- jeżeli dostępna szerokość zmniejszy się poniżej 769px, dodatkowo przepływ w kontenerze `main_container` zmieni się na pionowy oraz zostaną dostosowane parametry wyświetlania `mainitem_nav`. Cała strona będzie prezentowana w układzie pionowym.

```
110 @media screen and (max-width: 769px) {
111   .item_art {
112     -webkit-flex-direction: column;
113     flex-direction: column;
114   }
115   .main_container {
116     -webkit-flex-direction: column;
117     flex-direction: column;
118     align-items: stretch;
119     justify-content: stretch;
120     align-content: stretch;
121   }
122   .mainitem_nav {
123     flex: 1 1 auto;
124     width: 100%;
125     max-width: 769px;
126     padding: 20px;
127   }
128 }
```

Układ strony przy szerokości okna przeglądarki równej 900px oraz 700px (Firefox):



4.4 Układ oparty na CSS Grid layout

Układ typu CSS Grid Layout należy do najnowszych, jest dostępny od marca 2017 r. Początkowo obsługiwały go tylko: Chrome od wersji 57 i Firefox od wersji 52. Obecnie jest już wspierany przez wszystkie popularne przeglądarki. Jego możliwości są doceniane przez programistów, powszechnie jest uważany za jeden z najlepszych sposobów tworzenia układu elementów strony internetowej.

Jest to pierwszy układ w pełni 2-wymiarowy. Używając go należy zdefiniować wiersze i kolumny tworząc komórki, a następnie przypisać poszczególne elementy do komórek. Komórki można bardzo łatwo łączyć tworząc większe obszary, a także modyfikować te obszary w zależności np. od wielkości obszaru okna przeglądarki, co sprawia, że grid layout jest elastyczny i bardzo dobrze nadaje się do budowy responsywnych interfejsów użytkownika.

Należy po raz kolejny przejść na gałąź podstawową projektu (master / main), a następnie utworzyć nową gałąź o nazwie PPF_LAB2_GRID.

Pierwszym krokiem budowy przykładu praktycznego będzie modyfikacja pliku index.html. W tym przypadku, ponieważ grid layout jest z definicji 2-wymiarowy, nie potrzebujemy wielopoziomowego zagnieżdżania znaczników. Wszystkie obszary dokumentu będą definiowane znacznikami typu `<div>` i zostaną utworzone **na jednym poziomie**, wewnątrz głównego kontenera, którym będzie `<div>` klasy `main_page`. Natomiast treść zawarta wewnątrz modyfikowanych znaczników powinna pozostać niezmieniona. Poniżej został przedstawiony układ który należy uzyskać oraz zmiany nazw znaczników w porównaniu do dokumentu bazowego:

```
9      <body>
10     <div class="main_page">
11 >   <div class="div_head"> ...
16   </div>
17 >   <div class="div_nav"> ...
34   </div>
35 >   <div class="div_art_header"> ...
37   </div>
38 >   <div class="div_art_img1"> ...
40   </div>
41 >   <div class="div_art_text1"> ...
48   </div>
49 >   <div class="div_art_img2"> ...
51   </div>
52 >   <div class="div_art_text2"> ...
60   </div>
61 >   <div class="div_art_img3"> ...
63   </div>
64 >   <div class="div_art_text3"> ...
72   </div>
73 >   <div class="div_art_video"> ...
79   </div>
80 >   <div class="div_foot"> ...
85   </div>
86 </div>
87 </body>
```

Zmiany nazw i klas znaczników w porównaniu do dokumentu bazowego:

- `<div>` bezpośrednio wewn. `<body>` → `<div class="main_page">`
- `<header>` → `<div class="div_head">`
- `<nav>` → `<div class="div_nav">`
- `<div id="animals">` → `<div class="div_art_header" id=...>`
- `<div>` → `<div class="div_art_img1"><img ...`
- `<div><p>Żaba...` → `<div class="div_art_text1"><p>Ża...`
- `<div>` → `<div class="div_art_img2"><img...`
- `<div><p>Jeże...` → `<div class="div_art_text2"><p>Je...`
- `<div>` → `<div class="div_art_img3"><img ...`
- `<div><p>Biedronka...` → `<div class="div_art_text3"><p>Bi...`
- `<div>` zawierający `<h2>` i `<video>` → `<div class="div_art_video">`
- `<footer>` → `<div class="div_foot">`

Implementacja układu grid polega na zaprojektowaniu 2-wymiarowej siatki definiującej wiersze i kolumny, a następnie przypisaniu w pliku CSS poszczególnym klasom obszarów. W oparciu o zawartość dokumentu html, proponuję następującą siatkę podstawową:

	1	2	3	4
1	div_head	div_head	div_head	
2	div_nav	div_art_header	div_art_header	
3	div_nav	div_art_img1	div_art_text1	
4	div_nav	div_art_img1	div_art_text1	
5	div_nav	div_art_img2	div_art_text2	
6	div_nav	div_art_img2	div_art_text2	
7	div_nav	div_art_img3	div_art_text3	
8	div_nav	div_art_img3	div_art_text3	
9	div_nav	div_art_video	div_art_video	
10	div_nav	div_art_video	div_art_video	
11	div_foot	div_foot	div_foot	
12				

Warto zauważyć, że obszary nie są definiowane przez ich numer, ale przez numer linii oddzielającej. Numerowanie wierszy i kolumn rozpoczyna się od 1.

Uwaga 1:

Jak widać, podstawowa siatka w naszym przypadku dotyczy szerokiego ekranu (PC). Szczególne warianty będą definiowane dla mniejszych szerokości (tablet, smartphone), jak w poprzednich punktach.

Uwaga 2

Może się w tym momencie wydawać, że niektóre klasy (np. `div_art_img1`, `div_art_text1`, ..., `div_art_video`) niepotrzebnie zajmują po dwa wiersze. Powody się wyjaśnią, gdy za chwilę będziemy modyfikować układ dla mniejszych szerokości ekranu.

Uwaga 3

W praktyce układ grid jest często łączony z flexbox. W naszym projekcie zostanie to pokazane na przykładzie paska tytułowego, który będzie flexboxem zagnieżdżonym w komórce grid.

Plik style.css należy zdefiniować w następujący sposób:

- na początek kilka ogólnych właściwości, jak w poprzednich podpunktach:

```
1  * {
2  |   box-sizing: border-box;
3  | }
4  body {
5  |   font-family: Arial, Helvetica, sans-serif;
6  | }
7  h2 {
8  |   font-size: 20px;
9  |   width: 100%;
10 | }
11 p {
12 |   text-align: justify;
13 | }
```

- następnie najważniejsze ustawienie: definicja stylu wyświetlania grid dla klasy `main_page`. Dodatkowo należy określić liczbę kolumn i wierszy, stosując dowolne proporcje podziału. W naszym przypadku szerokość będzie dzielona w proporcjach: 2/2/6, (za pomocą jednostki `fraction of available space`) natomiast wysokości 11 wierszy będą definiowane automatycznie poprzez zawartość:

```
15 .main_page {
16 |   display: grid;
17 |   grid-template-columns: 2fr 2fr 6fr;
18 |   grid-template-rows: auto * 11;
19 | }
```

- element `div_head` ma zajmować cały pierwszy wiersz. Stąd jego szerokość (`grid-column`) jest definiowana od 1 do 4 linii siatki, a wysokość (`grid-row`) od 1 do 2 linii (patrz schemat siatki na poprzedniej stronie). Dodatkowo, zawartość tego elementu będzie typu flexbox:

```
21 .div_head {
22 |   grid-row: 1/2;
23 |   grid-column: 1/4;
24 |   display: flex;
25 |   flex-direction: row;
26 |   flex-wrap: wrap;
27 |   align-items: center;
28 |   justify-content: center;
29 |   width: 100%;
30 |   background-color: #c0c0c0;
31 |   padding: 10px;
32 | }
33 .div_head h1 {
34 |   color: #2e2e2e;
35 |   flex: 1 1 auto;
36 |   font-size: 24px;
37 |   text-align: center;
38 | }
```


- element `div_nav` będzie zajmował obszar od 2 do 10 wiersza z lewej strony:

```

40 .div_nav {
41     grid-row: 2/11;
42     grid-column: 1/2;
43     background-color: #cccccc;
44     padding: 20px;
45 }
46 .mainitem_nav ul {
47     list-style-type: none;
48     padding: 0;
49 }
50 .mainitem_nav li {
51     padding-bottom: 12px;
52 }

```

- następnie `div_art_header`: druga i trzecia kolumna w drugim wierszu:

```

54 .div_art_header {
55     grid-row: 2/3;
56     grid-column: 2/4;
57     padding: 10px;
58     background-color: #f1f1f1;
59 }

```

- każdy element z grafiką oraz z opisem będzie zajmował obszar składający się z dwóch wierszy i jednej kolumny. Poniżej przykład dla `div_art_img1` i `div_art_text1`. Pozostałe cztery, z numerami 2 i 3 należy zdefiniować w analogiczny sposób:

```

61 .div_art_img1 {
62     grid-row: 3/5;
63     grid-column: 2/3;
64     padding: 10px;
65     background-color: #f1f1f1;
66 }
67 .div_art_text1 {
68     grid-row: 3/5;
69     grid-column: 3/4;
70     padding: 10px;
71     background-color: #f1f1f1;
72 }

```

- pozostał jeszcze obszar odtwarzacza wideo w 2 wierszach i 2 kolumnach:

```

100 .div_art_video {
101     grid-row: 9/11;
102     grid-column: 2/4;
103     padding: 10px;
104     background-color: #f1f1f1;
105 }

```

- oraz stopka, w ostatnim wierszu na szerokości 3 kolumn:

```

107 .div_foot {
108     grid-row: 11/12;
109     grid-column: 1/4;
110     background-color: rgb(182, 182, 182);
111     padding: 10px;
112     text-align: center;
113     color: rgb(45, 3, 230);
114 }

```

W ten sposób zdefiniowaliśmy podstawowy układ oparty na grid. Teraz za pomocą media queries zostaną dodane modyfikacje dla ekranu o średniej szerokości (do 1024px) oraz małej szerokości (do 600px). Uwaga: ważna jest kolejność: najpierw `max-width: 1025px`, a dopiero potem `max-width: 601px`.

<pre>116 @media screen and (max-width: 1025px) { 117 .div_art_img1 { 118 grid-row: 3/4; 119 grid-column: 2/4; 120 } 121 .div_art_text1 { 122 grid-row: 4/5; 123 grid-column: 2/4; 124 } 125 .div_art_img2 { 126 grid-row: 5/6; 127 grid-column: 2/4; 128 } 129 .div_art_text2 { 130 grid-row: 6/7; 131 grid-column: 2/4; 132 } 133 .div_art_img3 { 134 grid-row: 7/8; 135 grid-column: 2/4; 136 } 137 .div_art_text3 { 138 grid-row: 8/9; 139 grid-column: 2/4; 140 } 141 }</pre>	<pre>143 @media screen and (max-width: 601px) { 144 .div_head { 145 grid-row: 1/2; 146 grid-column: 1/4; 147 } 148 .div_nav { 149 grid-row: 2/3; 150 grid-column: 1/4; 151 } 152 .div_art_header { 153 grid-row: 3/4; 154 grid-column: 1/4; 155 } 156 .div_art_img1 { 157 grid-row: 4/5; 158 grid-column: 1/4; 159 } 160 .div_art_text1 { 161 grid-row: 5/6; 162 grid-column: 1/4; 163 } 164 .div_art_img2 { 165 grid-row: 6/7; 166 grid-column: 1/4; 167 } 168 .div_art_text2 { 169 grid-row: 7/8; 170 grid-column: 1/4; 171 } 172 .div_art_img3 { 173 grid-row: 8/9; 174 grid-column: 1/4; 175 } 176 .div_art_text3 { 177 grid-row: 9/10; 178 grid-column: 1/4; 179 } 180 .div_art_video { 181 grid-row: 10/11; 182 grid-column: 1/4; 183 } 184 .div_foot { 185 grid-row: 11/12; 186 grid-column: 1/4; 187 } 188 }</pre>
---	--

CSS Grid Layout definiowany za pomocą GRID AREAS

Ostatnim zadaniem w ramach niniejszego laboratorium będzie zmiana sposobu definiowania obszarów w układzie grid layout. Zamiast numerów linii siatki możemy posłużyć się zdefiniowanymi nazwami i 2-wymiarową tabelą przypisującą te nazwy.

Proponuję na bazie gałęzi PPF_LAB2_GRID zdefiniować teraz jeszcze jedną nową gałąź PPF_LAB2_GRID_AREAS. Zmiany będą obejmowały tylko plik style.css:

- po pierwsze, ze wszystkich klas należy usunąć właściwości `grid-row` i `grid-column`, natomiast zamiast nich dodać właściwość `grid-area` i nadać jej wartość taką jak nazwa klasy z dodanym przyrostkiem `_area`. Np. w klasie `div_head` trzeba dodać następującą linię:

`grid-area: div_head_area;` Poniżej jako przykład podano definicję klasy `div_head`. W taki sam sposób należy zmodyfikować pozostałe klasy: `div_nav`, `div_art_header`, `div_art_img1`, `div_art_text1`, `div_art_img2`, `div_art_text2`, `div_art_img3`, `div_art_text3`, `div_art_video`, `div_foot`.

```
34 .div_head {
35     grid-area: div_head_area;
36     display: flex;
37     flex-direction: row;
38     flex-wrap: wrap;
39     align-items: center;
40     justify-content: center;
41     width: 100%;
42     background-color: #rgb(182, 182, 182);
43     padding: 10px;
44 }
```

- po zdefiniowaniu nazw obszarów można je przypisać w klasie kontenera `main_page`. Należy wprowadzić definicje nazw obszarów dla kolejnych wierszy. Najwygodniej ustawić wiersze pionowo, wtedy klasa kontenera ma następującą postać:

```
16 .main_page {
17     display: grid;
18     grid-template-columns: 2fr 2fr 6fr;
19     grid-template-rows: auto * 11;
20     grid-template-areas:
21     "div_head_area div_head_area div_head_area"
22     "div_nav_area div_art_header_area div_art_header_area"
23     "div_nav_area div_art_img1_area div_art_text1_area"
24     "div_nav_area div_art_img1_area div_art_text1_area"
25     "div_nav_area div_art_img2_area div_art_text2_area"
26     "div_nav_area div_art_img2_area div_art_text2_area"
27     "div_nav_area div_art_img3_area div_art_text3_area"
28     "div_nav_area div_art_img3_area div_art_text3_area"
29     "div_nav_area div_art_video_area div_art_video_area"
30     "div_nav_area div_art_video_area div_art_video_area"
31     "div_foot_area div_foot_area div_foot_area";
32 }
```

- Jedną z największych zalet stosowania obszarów za pomocą właściwości `grid-area` i `grid-template-areas` jest łatwość zmiany ułożenia poszczególnych elementów na stronie. Wystarczy tylko zmienić definicje szablonu `grid-template-areas` w poszczególnych *media queries*:

```

118 @media screen and (max-width: 1025px) {
119     .main_page {
120         grid-template-areas:
121             "div_head_area div_head_area div_head_area"
122             "div_nav_area div_art_header_area div_art_header_area"
123             "div_nav_area div_art_img1_area div_art_img1_area"
124             "div_nav_area div_art_text1_area div_art_text1_area"
125             "div_nav_area div_art_img2_area div_art_img2_area"
126             "div_nav_area div_art_text2_area div_art_text2_area"
127             "div_nav_area div_art_img3_area div_art_img3_area"
128             "div_nav_area div_art_text3_area div_art_text3_area"
129             "div_nav_area div_art_video_area div_art_video_area"
130             "div_nav_area div_art_video_area div_art_video_area"
131             "div_foot_area div_foot_area div_foot_area";
132     }
133 }

135 @media screen and (max-width: 601px) {
136     .main_page {
137         grid-template-areas:
138             "div_head_area div_head_area div_head_area"
139             "div_nav_area div_nav_area div_nav_area"
140             "div_art_header_area div_art_header_area div_art_header_area"
141             "div_art_img1_area div_art_img1_area div_art_img1_area"
142             "div_art_text1_area div_art_text1_area div_art_text1_area"
143             "div_art_img2_area div_art_img2_area div_art_img2_area"
144             "div_art_text2_area div_art_text2_area div_art_text2_area"
145             "div_art_img3_area div_art_img3_area div_art_img3_area"
146             "div_art_text3_area div_art_text3_area div_art_text3_area"
147             "div_art_video_area div_art_video_area div_art_video_area"
148             "div_art_video_area div_art_video_area div_art_video_area"
149             "div_foot_area div_foot_area div_foot_area";
150     }
151 }

```

Bibliografia:

- [1] Gajda W., HTML5 i CSS3 Praktyczne projekty, Helion 2013, ISBN 978-83-246-3050-9;
- [2] McFarland D.S., CSS3 nieoficjalny podręcznik, Helion 2013, ISBN 978-83-246-7317-9;
- [3] Kurs HTML w3schools: <https://www.w3schools.com/html/default.asp>
- [4] Kurs CSS w3schools: <https://www.w3schools.com/css/default.asp>