Opis Rozwiązania Zadania 2

Dominik Mistera

23 marca 2020

1 Wstęp

Celem zadania było stworzenie responsywnej strony internetowej która zmieniałaby rozmiar oraz zawartość w zależności od urządzenia na którym jest wyświetlana.

2 Sposób rozwiązania

Podpunkt 1

Treść

Zacznij od przygotowania samej struktury strony, czyli dokumentu html. Postaraj się (sensownie) wykorzystać jak najwięcej znaczników semantycznych. Na pierwszy rzut oka widać, że strona jest podzielona na trzy główne sekcje: nagłówek, zawartość oraz stopkę, które można zawrzeć odpowiednio w znacznikach header, main oraz footer. W nagłówku jest pole z napisem RS, które posłuży jako logo, dalej znajduje się tytuł strony oraz sekcja nawigacyjna (nav) składająca się z linków. Sekcja główna składa się z trzech artykułów (article), z których każdy ma nagłówek oraz akapit tekstu. Stopka może składać się np. z dwóch akapitów. Po uruchomieniu strona bez żadnych stylów powinna wyglądać mniej więcej tak jak na poniższym zrzucie ekranu.

Rozwiązanie

Podzieliłem sekcję body na 3 podsekcje: header, maain oraz footer. Poszczególne artykóły rozdzieliłem znacznikiem article. Kazdy artykuł ma tytuł umieszczony w znaczniku h3 oraz treść umieszczoną w paragrafie. W nagłówku umieściłem logo w znaczniku img, tytuł główny strony w znaczniku h1 oraz linki do nawigacji w znaczniku nav.

```
<html>
    <head>
        <title>Title</title>
    </head>
    <body>
            <header>
                <div id="logo-container">
                    <img id="logo" src="logo.jpg">
                </div>
```

```
                <h1 id="title">Responsywna strona</h1>
                <nav>
                    <a>[...]</a>
                    [...]
                    <a>[..]</a>
                </nav>
            </header>
            <main>
                <article>
                    <h3>Header1</h3>
                    <p>[...]</p>
                </article>
                <article>
                    <h3>Header2</h3>
                    <p>[...]</p>
                </article>
                <article>
                    <h3>Header3</h3>
                    <p>[...]</p>
                </article>
            </main>
        <footer>
            <p>Projekt i przygotownaie</p>
            <p id="author">Dominik Mistera</p>
        </footer>
    </body>
</html>
```

Podpunkt 2

Treść

Zmianę wyglądu zaczniemy od zresetowania domyślnych marginesów (margin) i wypełnienia (padding) elementów html, body oraz p, ponieważ domyślnie mogą one być różne od 0, w zależności od przeglądarek. Utwórz osobny plik CSS, dołącz go do Twojej strony i wprowadź odpowiednią regułę.

Rozwiązanie

Należy ustawić właściwość padding oraz margin na 0 w każdym z wymienionych elementów:

```
html {
    padding: 0;
    margin: 0;
}

body {
    padding: 0;
    margin: 0;
}
```

```
p {
    padding: 0;
    margin: 0;
}
```

Podpunkt 3

Treść

Na początku warto również podjąć decyzję w jaki sposób będziemy definiowali wielkość elementów znajdujących się na stronie. Za to zadanie odpowiada w CSS właściwość box-sizing. Ustaw ją na jedną z wybranych przez Ciebie wartości dla wszystkich elementów w dokumencie. W wyborze może pomóc Tobie artykuł na stronie https://css-tricks.com/box-sizing/.

Rozwiązanie

Do zmiany wszystkich elementów mozna użyc selektora gwiazdka. Ustawiłem box-sizing na border-box dla wszystkich elementów:

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

Podpunkt 4

Treść

Ustaw wszystkim elementom czcionkę Verdana w rozmiarze 16px. Wyłącz również domyślne pogrubienie (font-weight) i podkreślenie (text-decoration) elementów, a także ustaw kolor wszystkich elementów na czarny

Rozwiązanie

Do zmiany wszystkich elementów mozna użyc selektora gwiazdka.

```
* {
    box-sizing: border-box;
    font-family: Verdana;
    font-size: 16px;
    font-weight: normal;
    text-decoration: none;
    color: black;
}
```

Podpunkt 5

Treść

Aby odpowiednio rozmieścić elementy na stronie posłużymy się głównie mechanizmem układu CSS zwanym flexbox. Wejdź na stronę https://css-tricks.com/snippets/css/a-guide-to-flexbox/ i zapoznaj się pokrótce z zamieszczonym tam przewodnikiem.

Dla lepszego zrozumienia i zapamiętania zaprezentowanych tam treści możesz również przejść grę znajdującą się na stronie https://flexboxfroggy.com, która w przystępny sposób zapozna Cię z głównymi mechanizmami flexbox'a. Zacznijmy od zapewnienia poprawnego układu w trzech głównych sekcjach naszej strony (nagłówek, treść, stopka). Korzystając z właściwości display oraz flex-direction ustaw parametry każdej z tych sekcji. Po tym zabiegu strona powinna wyglądać mniej więcej tak jak na poniższym zrzucie ekranu.

Rozwiązanie

Należy kontener tych trzech głownych sekcji na flexbox. Jest nim element body. Następnie aby każdy artykuły były ustawione obok siebie należy ustawić sekcje main na flexbox.

```
body {
    display: flex;
    flex-direction: column;
}

main {
    display: flex;
    flex-direction: row;
}
```

Podpunkt 6

Treść

Zaczniemy teraz sukcesywnie dostosowywać do naszych potrzeb każdą sekcję po kolei, od góry do dołu, zaczynając od nagłówka. Zacznij od dodania wypełnienia do nagłówka w wielkości 30px oraz wyrównania elementów w pionie (korzystając z jednej właściwości flexbox'a). Po tym zabiegu nagłówek powinien wyglądać tak.

Rozwiązanie

Należy zmodyfikować header w nastepujący sposób:

```
header {
    display: flex;
    flex-direction: row;
    align-items: center;
    padding: 30px;
}
```

Podpunkt 7

Treść

Dla tytułu strony (napis „Responsywna strona") w nagłówku ustaw lewy margines wielkości 15px oraz pogrubioną czcionkę w rozmiarze dwa razy większym niż na całej stronie.

Rozwiązanie

Należy odnieść się do tytułu poprzez identyfikator:

```
#title {
    font-size: 32px;
    margin-left: 15px;
}
```

Podpunkt 8

Treść

Spraw, aby sekcja z odnośnikami (nav) była (jeśli to możliwe) 2 razy większa niż sekcje z logo oraz z tytułem strony (flex-grow) a także przesuń wszystkie odnośniki w tej sekcji do górnej krawędzi sekcji (align-self). W sekcji nav również włącz mechanizm flexbox'a (display) a następnie przesuń wszystkie odnośniki na jej prawą stronę (justify-content).

Rozwiązanie

Używając flex grow można sprawić że sekcja nav będzie 2 razy większa od sekcji z tytułem. Następnine należy sprawić aby element nav stał się flex kontenerem. Aby przesunąć wszystkie elementy na prawo należy je przesunąć na koniec (flex-end).

```
#title {
    flex-grow: 1;
}

nav {
    flex-grow: 2;
    align-self: flex-start;
    flex-direction: row;
    justify-content: flex-end;
    display: flex;
}
```

Podpunkt 9

Treść

Ustaw szerokość każdego odnośnika na 75px.

Rozwiązanie

Podpunkt 10

Treść

W miejscu logo (zamiast napisu „RS") wstaw obrazek lub samodzielnie stwórz w CSS logo o wymiarach 75px na 75px.

Rozwiązanie

Aby obrazek był przeskalowany do wymiarów 75 na 75 umieściłem go w kontenerze o danych wymiarach nastepnie sam obrazek rozciągnąłem aby wypełnił cały kontener.

```
#logo-container {
    width: 75px;
    height: 75px;
}

#logo {
    max-width: 100%;
    max-height: 100%;
}
```

Podpunkt 11

Treść

Po powyższych zabiegach cała strona powinna wyglądać mniej więcej następująco.

Podpunkt 12

Treść

Teraz zabierzemy się za sekcję główną. Chcemy, żeby odstępy od artykułów do krawędzi, jak również odstępy między samymi artykułami, wynosiły tyle samo ile w nagłówku, czyli 30px. Efekt ten osiągniemy składając wypełnienie sekcji głównej oraz marginesy artykułów. Ustaw obie te wartości na 15px, co łącznie da nam 30px.

Rozwiązanie

Należy zmienić elementy main oraz article:

```
main {
    padding: 15px;
}

article {
    margin: 15px;
}
```

Podpunkt 13

Treść

Aby każdy artykuł zajmował tyle samo miejsca, niezależnie od ilości zawartego w nim tekstu, ustaw szerokość każdego artykułu na 100o grubości 2px.

Rozwiązanie

```
article {
    width: 100%;
    border-width: 2px;
    border-style: solid;
}
```

Podpunkt 14

Treść

Zajmiemy się teraz nagłówkami w artykułach. Zacznij od ustawienia każdemu nagłówkowi wysokości 50px, szerokości 50% oraz rozmiaru czcionki 1.5em, a następnie wycentruj tekst w każdym nagłówku (display, justify-content, align-items).

Rozwiązanie

Do nagłówna artykułu można się dostać stosowając selektor article h3. Należy zmienić układ na flexbox.

```
article h3 {
    height: 50px;
    width: 50%;
    font-size: 1.5em;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

Podpunkt 15

Treść

Teraz wyrównamy nagłówki do prawej strony. Można to osiągnąć na wiele sposobów, jednak w myśl zasady, że jak się ma dobry młotek to wszystko wygląda jak gwóźdź, my posłużymy się oczywiście flexbox'em. Zacznij od włączenia układu flexbox i ustawienia kierunku kolumnowego w artykułach, a następnie w nagłówkach ustaw odpowiednie wyrównanie (align-self).

Rozwiązanie

Ustawiłem w nagłowku właściwość align-self na flex-end ponieważ jest to prawa strona znajduje się na końcu w układzie kolumnowym.

```
article {
    display: flex;
    flex-direction: column;
}

article h3 {
    align-self: flex-end;
```

```
}
```

Podpunkt 16

Treść

Teraz pozostaje tylko ustawić w nagłówkach jasnoszare tło oraz lewą i dolną czarną ramkę o grubości 2px.

Rozwiązanie

```
article h3 {
    background-color: lightgray;
    border-left-width: 2px;
    border-bottom-width: 2px;
    border-left-style: solid;
    border-bottom-style: solid;
    margin: 0;
}
```

Podpunkt 17

Treść

Ostatnim zabiegiem pozostaje wyjustowanie akapitów w artykułach (text-align) oraz ustawienie im wypełnienia o wielkości 15px.

Rozwiązanie

```
article p {
    padding: 15px;
    text-align: justify;
}
```

Podpunkt 18

Treść

Teraz pozostaje nam jedynie zająć się stopką. Tutaj sprawa jest już relatywnie prosta. Ustaw stopce stałą wysokość 100px, jasnoszare tło, czarną górną ramkę o grubości 2px i górny margines o wysokości 15px, a następnie wycentruj zawartość w pionie i w poziomie (align-items, justify-content). Dodatkowo, ustaw drugiemu akapitowi w stopce pogrubioną czcionkę. Efekt powinien być następujący

Rozwiązanie

Odniosłem się do drugiego akapitu za pomocą identyfikatora.

```
footer {
    height: 100px;
    background-color: lightgray;
    border-top-width: 2px;
```

```
    border-top-style: solid;
    margin-top: 15px;
    align-items: center;
    justify-content: center;
    display: flex;
    flex-direction: column;
}

#author {
    font-weight: bold;
}
```

Podpunkt 19

Treść

. Jest już prawie dobrze, ale pozostaje jeszcze rozwiązać kwestię pozycji stopki, która powinna być zawsze przyklejona do dołu strony (tzw. sticky footer). Jak ze wszystkim innym w CSS i tutaj jest wiele opcji rozwiązania tego problemu (https://css-tricks.com/couple-takes-sticky-footer/). Skorzystaj z jednej z nich (na tym etapie chyba najłatwiej będzie... flexbox'em). Zauważ, że niestety nie obejdzie się bez dodatkowego elementu (div z klasą content), w którym zawrzemy wszystko poza stopką. Efekt końcowy powinien być następujący.

Rozwiązanie

Należy umieścić sekcję header oraz main w kontenerze (nazwałem go content). Następnie używając flex-grow oraz flex-shrink sprawiłem aby stopka znajdowała się zawsze na dole.

```
#content {
    flex-grow: 1;
}

footer {
    flex-shrink: 0;
}
```

Podpunkt 20

Treść

. Na tym etapie mamy gotową stronę w wersji na duże ekrany. Teraz pozostaje nam obsłużyć małe ekrany. Aby móc definiować style w zależności od parametrów okna przeglądarki posłużymy się zapytaniami media w CSS. Zaczniemy od sekcji głównej, bo tutaj sprawa jest relatywnie prosta. Na końcu swojego arkusza stylów dodaj poniższą regułę i zobacz co dzieje się ze stroną kiedy przechodzisz przez próg 900px szerokości okna przeglądarki.

Rozwiązanie

Gdy strona przechodzi przez próg 900px, artykuły w sekcji głownej przeskakują do nowego rzzędu.

Podpunkt 21

Treść

Teraz dodaj do zapytania media reguły, które sprawią, że tytuł strony oraz sekcja nawigacyjna znikną na mniejszych urządzeniach (display: none), a w ich miejsce pojawi się tzw. „hamburger", czyli ikona zwiniętego menu. Dodaj w sekcji nagłówkowej strony jako ostatni element „hamburgera" (możesz dodać obrazek lub spróbować stworzyć go samodzielnie z elementów HTML i reguł CSS) i ustaw go domyślnie jako niewyświetlany, a na mniejszych urządzeniach jako wyświetlany (display: block). Dodatkowo zmodyfikuj na mniejszych urządzeniach sposób rozmieszczenia elementów w sekcji nagłówkowej tak, aby logo pozostało po lewej stronie strony, a hamburger był po prawej (justify-content). Efekt końcowy na małych urządzeniach powinien być następujący.

Rozwiązanie

Do sekcji header dodałem nowy element hamburgera:

```
<div id="hamburger-container">
  <img id="hamburger" src="hamburger.png">
</div>
```

Następnie ustawiłem wielkośc tego elementu w taki sam sposób jak logo:

```
#hamburger-container {
    width: 75px;
    height: 75px;
    display: none;
}

#hamburger {
    max-width: 100%;
    max-height: 100%;
}
```

Następnie dodałem reguły które aktywują się gdy strona ma szerokośc mniejszą niż 900px. Ukryłem sekcje nawigacyjną oraz pokazałem element hamburgera modyfikując właściwość display. Przesunąłem element hamburgera na prawą strone za pomocą własciwości justify-content:

```
@media all and (max-width: 900px) {
    main {
        flex-wrap: wrap;
    }

    #title {
        display: none;
    }
```

```
    nav {
        display: none;
    }

    #hamburger-container {
        display: block;
    }

    header {
        justify-content: space-between;
    }
}
```

3 Podsumowanie

W rezultacie uzyskałem responsywną stronę która zmienia swój wygląd w zależności od szerokości przeglądarki. Dzięki temu strona powinna wygladać przejrzyście na urządzeniach mobilnych.