Projektowanie Aplikacji Internetowych
Wykład 3

Stylowanie i układ strony

Mateusz Pawełkiewicz

1. Podstawy składni CSS

1.1 Czym jest CSS

CSS (Cascading Style Sheets) to język arkuszy stylów, który pozwala na kontrolowanie wyglądu i formatowania elementów HTML na stronie internetowej. Umożliwia oddzielenie treści (HTML) od prezentacji (CSS), co zwiększa elastyczność i ułatwia zarządzanie stronami WWW.

1.2 Reguły CSS

Reguła CSS składa się z selektora oraz bloku deklaracji:

```
selektor {
  właściwość: wartość;
}
```

- **Selektor**: Określa, do jakich elementów HTML zostanie zastosowany styl.
- **Blok deklaracji**: Zawiera pary właściwość-wartość, określające stylizację.

1.3 Właściwości i wartości

- **Właściwości**: Atrybuty stylu, takie jak color, font-size, margin.
- **Wartości**: Ustawienia dla danej właściwości, np. red, 16px, auto.

1.4 Komentarze w CSS

Komentarze są ignorowane przez przeglądarki i służą do dodawania notatek w kodzie.

```
/* To jest komentarz w CSS */
```

1.5 Sposoby dołączania CSS do HTML

- **Wewnętrzne style (inline styles)**:

  ```
  <p style="color: blue;">Tekst</p>
  ```

- **Style w sekcji <style> w nagłówku**:

  ```
  <head>
    <style>
      p {
        color: blue;
      }
    </style>
  </head>
  ```

- **Zewnętrzny arkusz stylów**:
  - W pliku CSS (np. style.css):

```
p {
  color: blue;
}
```

  - W pliku HTML:

```
<head>
  <link rel="stylesheet" href="style.css" />
</head>
```

---

2. Selektory, klasy i identyfikatory

2.1 Selektory

Selektory określają, do jakich elementów zostaną zastosowane style.

2.1.1 Selektory elementów (tagów)

Stylizacja wszystkich elementów danego typu.

```
p {
  font-size: 16px;
}
```

2.1.2 Selektory klas

Używane do stylizacji elementów z określoną klasą. Klasy definiuje się w HTML za pomocą atrybutu class.

- W CSS: poprzedzone kropką .

```
.nazwa-klasy {
  color: red;
}
```

- W HTML:

```
<p class="nazwa-klasy">Tekst</p>
```

2.1.3 Selektory identyfikatorów

Stylizacja elementu o unikalnym identyfikatorze. Identyfikatory definiuje się w HTML za pomocą atrybutu id.

- W CSS: poprzedzone krzyżykiem #

```
#nazwa-id {
  background-color: yellow;
}
```

- W HTML:

```
<div id="nazwa-id">Zawartość</div>
```

2.1.4 Selektory atrybutów

Stylizacja elementów na podstawie ich atrybutów.

```
input[type="text"] {
  border: 1px solid #ccc;
}
```

2.1.5 Selektory złożone

- **Potomny**: Wybiera elementy wewnątrz innego elementu.

```
ul li {
  list-style-type: none;
}
```

- **Bezpośredni dziecko**: Wybiera elementy będące bezpośrednimi dziećmi innego elementu.

```
div > p {
  color: green;
}
```

- **Sąsiedni**: Wybiera elementy następujące bezpośrednio po innym elemencie.

```
h2 + p {
  margin-top: 10px;
}
```

2.1.6 Pseudoklasy

Stylizacja elementów w określonym stanie.

- **Pseudoklasy linków**: :link, :visited

```
a:link {
  color: blue;
}
a:visited {
  color: purple;
}
```

- **Pseudoklasy interaktywne**: :hover, :active, :focus

```
button:hover {
  background-color: lightblue;
}
```

- **Pseudoklasy strukturalne**: :first-child, :last-child, :nth-child(n)

```
li:first-child {
  font-weight: bold;
}
```

2.1.7 Pseudoelementy

Stylizacja części elementu.

- **::before i ::after**

```
p::first-letter {
  font-size: 2em;
}
```

---

3. Model pudełkowy (Box Model)

Model pudełkowy opisuje, jak przeglądarka oblicza szerokość, wysokość i przestrzeń wokół elementów HTML.

3.1 Składowe modelu pudełkowego

- **Content (Zawartość)**: Obszar, w którym wyświetlana jest treść (tekst, obrazki).
- **Padding (Wypełnienie)**: Przestrzeń między zawartością a obramowaniem.
- **Border (Obramowanie)**: Linie otaczające element.
- **Margin (Margines)**: Przestrzeń na zewnątrz obramowania, oddzielająca element od innych.

3.2 Obliczanie rozmiarów

Domyślnie, całkowita szerokość i wysokość elementu to:

- **Szerokość całkowita**:

  width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

- **Wysokość całkowita**:

  height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

3.3 Właściwości CSS związane z modelem pudełkowym

- **width i height**: Określają wymiary zawartości elementu.
- **padding**:

  ```
  padding: 10px; /* wszystkie krawędzie */
  padding: 10px 20px; /* góra/dół, lewo/prawo */
  padding: 10px 15px 20px 25px; /* góra, prawo, dół, lewo */
  ```

- **border**:

  ```
  border: 1px solid #000; /* szerokość, styl, kolor */
  ```

- **margin**: Podobne do padding, określa marginesy wokół elementu.

3.4 box-sizing

Właściwość box-sizing pozwala zmienić sposób obliczania szerokości i wysokości elementu.

- **content-box** (domyślne): width i height odnoszą się do zawartości.
- **border-box**: width i height obejmują zawartość, padding i border.

Przykład ustawienia dla całego dokumentu:

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

---

4. Techniki układu strony

4.1 Flexbox

Flexbox (Flexible Box Layout) to moduł CSS, który ułatwia tworzenie elastycznych układów w jednym wymiarze (rzędzie lub kolumnie).

4.1.1 Podstawy Flexboxa

- Ustawienie na kontenerze:

  ```
  .container {
    display: flex;
  }
  ```

- **Główne właściwości kontenera**:
  - flex-direction: Określa kierunek osi głównej (row, row-reverse, column, column-reverse).
  - justify-content: Wyrównanie wzdłuż osi głównej (flex-start, flex-end, center, space-between, space-around, space-evenly).
  - align-items: Wyrównanie wzdłuż osi poprzecznej (stretch, flex-start, flex-end, center, baseline).
  - flex-wrap: Czy elementy mają zawijać się na kolejne linie (nowrap, wrap, wrap-reverse).

4.1.2 Właściwości elementów potomnych

- flex-grow: Określa, jak elementy rosną wzdłuż osi głównej.
- flex-shrink: Określa, jak elementy kurczą się wzdłuż osi głównej.
- flex-basis: Podstawowa wielkość elementu przed rozdzieleniem dostępnej przestrzeni.
- Skrót flex:

  ```
  flex: flex-grow flex-shrink flex-basis;
  ```

- align-self: Nadpisuje align-items dla konkretnego elementu.

4.1.3 Przykład zastosowania

HTML:

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
</div>
```

CSS:

```
.container {
  display: flex;
  justify-content: space-between;
}

.item {
  flex: 1;
  margin: 5px;
}
```

4.2 CSS Grid

CSS Grid Layout to dwuwymiarowy system układu, który pozwala tworzyć złożone siatki.

4.2.1 Podstawy CSS Grid

- Ustawienie na kontenerze:

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

- **Definiowanie kolumn i wierszy**:

```
.grid-container {
  grid-template-columns: 1fr 1fr 1fr; /* trzy kolumny o równej szerokości */
  grid-template-rows: auto 200px; /* dwa wiersze */
}
```

- **Odstępy między elementami**:

```
grid-gap: 10px; /* odstęp między wierszami i kolumnami */
```

4.2.2 Pozycjonowanie elementów

- **Automatyczne rozmieszczenie**: Elementy wstawiane są automatycznie w wolne miejsca.
- **Określanie pozycji elementów**:

```
.item1 {
  grid-column: 1 / 3; /* od kolumny 1 do 3 (zajmuje dwie kolumny) */
  grid-row: 1 / 2; /* wiersz 1 */
}
```

- **Nazwy linii siatki**:

```
.grid-container {
  grid-template-columns: [start] 1fr [middle] 1fr [end];
}
```

4.2.3 Przykład zastosowania

HTML:

```
<div class="grid-container">
  <div class="item1">Nagłówek</div>
  <div class="item2">Menu</div>
  <div class="item3">Treść</div>
  <div class="item4">Stopka</div>
</div>
```

CSS:

```
.grid-container {
  display: grid;
  grid-template-areas:
    "header header"
    "menu content"
    "footer footer";
  grid-gap: 10px;
}

.item1 {
  grid-area: header;
}
```

```
.item2 {
  grid-area: menu;
}

.item3 {
  grid-area: content;
}

.item4 {
  grid-area: footer;
}
```

4.3 Porównanie Flexbox i CSS Grid

- **Flexbox**:
  - Najlepszy do układów w jednym wymiarze (rzędy lub kolumny).
  - Elementy są rozmieszczane wzdłuż osi głównej.
- **CSS Grid**:
  - Zaprojektowany do układów dwuwymiarowych (rzędy i kolumny).
  - Pozwala na precyzyjne pozycjonowanie elementów.

---

5. Responsywność i projektowanie mobilne

5.1 Media queries

Media queries pozwalają na stosowanie stylów w zależności od właściwości urządzenia, takich jak szerokość ekranu, rozdzielczość czy orientacja.

5.1.1 Składnia media queries

```
@media (warunek) {
  /* Style */
}
```

5.1.2 Przykłady użycia

- **Styl dla urządzeń o szerokości ekranu do 600px**:

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

- **Styl dla urządzeń o szerokości ekranu od 601px do 1024px**:

```
@media (min-width: 601px) and (max-width: 1024px) {
  body {
    background-color: lightgreen;
  }
}
```

- **Styl dla urządzeń w orientacji pionowej**:

```
@media (orientation: portrait) {
  body {
    font-size: 14px;
  }
}
```

5.2 Mobile First

Strategia projektowania, w której najpierw tworzy się styl dla urządzeń mobilnych, a następnie rozszerza go dla większych ekranów.

- **Przykład**:

```
/* Styl podstawowy dla urządzeń mobilnych */
body {
  font-size: 14px;
}

/* Dodatkowe style dla większych ekranów */
@media (min-width: 768px) {
  body {
    font-size: 16px;
  }
}
```

5.3 Jednostki względne i bezwzględne

- **Jednostki bezwzględne**: px (piksele)
- **Jednostki względne**:
  - em: Względem rozmiaru czcionki rodzica
  - rem: Względem rozmiaru czcionki elementu html
  - vw, vh: Procent szerokości (vw) lub wysokości (vh) widocznego obszaru
- **Przykład użycia jednostek względnych**:

```
body {
  font-size: 16px; /* Ustawienie bazowego rozmiaru czcionki */
}

h1 {
  font-size: 2rem; /* 32px */
}
```

6. Animacje i przejścia w CSS

6.1 Przejścia (Transitions)

Przejścia pozwalają na płynne zmiany wartości właściwości CSS w określonym czasie.

6.1.1 Składnia przejść

```
element {
  transition: właściwość czas funkcja;
}
```

- **właściwość**: Nazwa właściwości, która ma ulegać zmianie (color, background-color, all).
- **czas**: Czas trwania przejścia (np. 0.5s).
- **funkcja**: Funkcja timingowa (ease, linear, ease-in, ease-out, ease-in-out).

6.1.2 Przykład użycia

```
.button {
  background-color: blue;
  transition: background-color 0.3s ease;
}

.button:hover {
  background-color: red;
}
```

6.2 Animacje (Animations)

Animacje umożliwiają tworzenie złożonych efektów poprzez definiowanie klatek kluczowych.

6.2.1 Definiowanie animacji

- **@keyframes**: Definiuje klatki kluczowe animacji.

```
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
```

- **Stosowanie animacji do elementu**:

```
.element {
  animation-name: fadeIn;
  animation-duration: 2s;
  animation-timing-function: ease-in;
  animation-delay: 0s;
  animation-iteration-count: 1;
  animation-fill-mode: forwards;
}
```

6.2.2 Skrócona składnia animacji

```
.element {
  animation: fadeIn 2s ease-in 0s 1 forwards;
}
```

- Składnia: animation: nazwa czas funkcja delay liczba-powtórzeń tryb-wypełnienia;

6.2.3 Przykład złożonej animacji

```
@keyframes slideIn {
  0% {
    transform: translateX(-100%);
  }
  50% {
    transform: translateX(10%);
  }
  100% {
    transform: translateX(0);
  }
}

.element {
  animation: slideIn 1s ease-out;
}
```

6.3 Transforms (Przekształcenia)

Pozwalają na obracanie, skalowanie, przesuwanie i pochylanie elementów.

- **Przykłady**:

```
.rotate {
  transform: rotate(45deg);
}

.scale {
  transform: scale(1.5);
}

.translate {
  transform: translate(50px, 100px);
}
```

```
.skew {
  transform: skew(20deg, 10deg);
}
```

6.4 Transition vs Animation

- **Transition**:
  - Umożliwia animowanie przejścia między dwoma stanami.
  - Wymaga zdarzenia inicjującego (np. :hover).
- **Animation**:
  - Pozwala na tworzenie złożonych animacji z wieloma klatkami kluczowymi.
  - Może być uruchamiana automatycznie.