

Podstawy informatyki

HTML5 / CSS3

Laboratorium 8

Model pudełkowy i właściwość display

Czas trwania: 2 × 45 min**Tryb:** studia stacjonarne**Jednostka:** Katedra Elektrotechniki i Technologii Nadprzewodnikowych**Prowadzący:** dr inż. Michał Łanczont

Cel ćwiczenia

Celem jest zrozumienie modelu pudełkowego CSS, różnicy `content-box`/`border-box`, odstępów, zjawiska zapadania marginesów oraz zachowania elementów `block`, `inline`, `inline-block`, `none`, `flex` i `grid`.

Wstęp

Każdy element renderowany przez CSS jest przedstawiany jako pudełko. Model pudełkowy opisuje relacje pomiędzy zawartością, dopełnieniem, obramowaniem i marginesem.

Warstwy modelu pudełkowego

1. **content** – treść i podstawowy obszar zawartości,
2. **padding** – odstęp pomiędzy treścią a obramowaniem,
3. **border** – obramowanie,
4. **margin** – zewnętrzny odstęp od sąsiednich elementów.

Przy domyślnym `box-sizing: content-box` deklarowana `width` odnosi się do obszaru zawartości. `Padding` i `border` zwiększają wtedy całkowity rozmiar elementu. Przy `border-box` deklarowana szerokość obejmuje również `padding` i `border`.

```
.box-a {  
  box-sizing: content-box;  
  width: 300px;  
  padding: 20px;  
  border: 5px solid #900;  
}  
  
.box-b {  
  box-sizing: border-box;  
  width: 300px;  
  padding: 20px;  
  border: 5px solid #090;  
}
```

Globalny border-box

`box-sizing` nie dziedziczy się automatycznie z `body`, dlatego praktyczny reset obejmuje wszystkie elementy i pseudoelementy:

```
*,
*::before,
*::after {
  box-sizing: border-box;
}
```

Padding, margin, border i skróty

```
.element {
  padding: 10px 20px;          /* pion | poziom */
  margin: 5px 10px 15px 20px; /* gora | prawo | dol | lewo */
  border: 2px solid blue;
  border-radius: 8px;
}
```

`margin: 0 auto` może wyśrodkować blok o szerokości mniejszej niż szerokość kontenera w klasycznym układzie blokowym.

Przykład wyróżnionego bloku

```
.cytat {
  width: 70%;
  border: 4px dashed #999;
  padding: 20px;
  margin: 30px auto;
  background-color: #fffacd;
}
```

display: block, inline, inline-block, none

Element o zewnętrznym trybie `block` zaczyna nowy wiersz i respektuje `width/height`. Zwykle pudełko `inline` pozostaje w linii; `width`, `height` oraz marginesy góra/dół nie wpływają na jego układ tak jak w bloku. `padding` i `border` pionowy mogą być narysowane, ale nie przesuwają sąsiednich linii w taki sam sposób i mogą powodować nakładanie.

```
span.blok {
  display: block;
  width: 100px;
  height: 50px;
}

.kafelek {
  display: inline-block;
  width: 30%;
  padding: 10px;
  border: 1px solid #999;
}

#reklama {
  display: none;
}
```

Elementy zastępowane

Nie wszystkie elementy o zachowaniu liniowym reagują identycznie. Obraz `img` jest elementem zastępowanym (*replaced element*) i może respektować `width/height`, mimo że uczestniczy w układzie liniowym. Dlatego zachowania `span` nie należy bezmyślnie przenosić na `img`.

Zapadanie marginesów

W normalnym układzie blokowym pionowe marginesy sąsiednich bloków mogą się “zapadać” (margin collapsing), zamiast po prostu sumować. Zjawisko nie dotyczy w ten sam sposób elementów flex/grid. Warto je rozpoznać, gdy dwa pionowe marginesy dają mniejszy odstęp niż oczekiwano.

Przykład obliczania szerokości

Dla `content-box` element o `width: 300px`, paddingu `20px` z obu stron i obramowaniu `5px` ma zewnętrzną szerokość:

$$300 + 2 \cdot 20 + 2 \cdot 5 = 350 \text{ px.}$$

Przy `border-box` te same padding i border są wliczane w deklarowane `300px`, więc obszar content staje się odpowiednio węższy.

```
.card {  
  box-sizing: border-box;  
  width: min(100%, 30rem);  
  padding: 1.25rem;  
  border: 2px solid #1f4e79;  
  margin: 1.5rem auto;  
}
```

Typowe błędy modelu pudełkowego

- ustawienie `box-sizing` tylko na `body` i oczekiwanie dziedziczenia,
- sumowanie pionowych marginesów bez uwzględnienia margin collapsing,
- próba ustawienia `width/height` na zwykłym `span` bez zmiany `display`,
- stałe szerokości powodujące overflow na małych ekranach,
- używanie paddingu jako substytutu odstępu pomiędzy osobnymi komponentami, gdzie właściwszy jest `margin/gap`.

Przebieg ćwiczenia

Zadanie 1. content-box kontra border-box

Do wykonania

Utwórz dwa identyczne `div` o `width: 400px`, `padding: 20px` i `border: 5px solid`. Pierwszemu pozostaw `content-box`, drugiemu ustaw `border-box`. Zmierz ich rzeczywistą szerokość w DevTools i oblicz ją ręcznie.

Zadanie 2. Globalny box-sizing

Do wykonania

Dodaj reset `*`, `::before`, `::after` { `box-sizing: border-box`; }. Sprawdź ponownie szerokości. Wyjaśnij, dlaczego samo ustawienie `box-sizing` na `body` nie rozwiązuje problemu dla wszystkich elementów.

Zadanie 3. Marginesy asymetryczne i centrowanie

Do wykonania

Utwórz `.pudelko-2` z różnymi marginesami: 10px, 20px, 40px, 80px. Utwórz `.centralny` o szerokości 50% i wyśrodkuj go przez `margin: 0 auto`. Następnie przygotuj dwa sąsiednie akapity z dużymi marginesami pionowymi i zaobserwuj margin collapsing.

Zadanie 4. Inline i inline-block

Do wykonania

Dla zwykłego `span` ustaw `width: 100px`, `height: 50px`, pionowy padding i pionowy margin. Zaobserwuj efekt. Następnie przełącz `display: inline-block` i porównaj zachowanie.

Zadanie 5. Kafelki inline-block

Do wykonania

Utwórz cztery elementy listy jako `inline-block` o szerokości 20% i paddingu 10px. Sprawdź wpływ białych znaków w HTML i zwężania okna. W komentarzu napisz, dlaczego dla wspólnego układu wielu kafelków zwykle wygodniejszy jest Flexbox lub Grid.

Zadanie 6. display none

Do wykonania

Dodaj blok `#reklama` i ustaw `display: none`. Sprawdź w DevTools, że element pozostaje w DOM, ale nie jest renderowany i nie zajmuje miejsca. Porównaj koncepcyjnie z `visibility: hidden` (bez konieczności stosowania w zadaniu).

Zadanie końcowe

Integracja

Zbuduj sekcję “karty produktu” demonstrującą padding, margin, border, `border-box`, element liniowy i liniowo-blokowy. Do każdej decyzji dodaj krótki komentarz CSS wyjaśniający jej wpływ na model pudełkowy.

Kryteria zaliczenia

- poprawne obliczanie szerokości content-box i border-box,
- globalne ustawienie box-sizing obejmujące pseudoelementy,
- rozpoznanie margin collapsing,

- poprawne wyjaśnienie inline vs inline-block,
- rozróżnienie zwykłego inline od replaced element, np. `img`.

Sprawozdanie

Do odbioru należy przygotować pliki źródłowe utworzone podczas ćwiczenia. Jeżeli prowadzący wymaga sprawozdania, powinno ono zawierać: krótki opis rozwiązania, najważniejsze fragmenty kodu, zrzut ekranu wyniku oraz wnioski dotyczące napotkanych problemów. Pliki HTML i CSS należy zachować w strukturze katalogów zgodnej z instrukcją.

Materiały pomocnicze

- HTML Living Standard: <https://html.spec.whatwg.org/>
- MDN Web Docs – HTML: <https://developer.mozilla.org/docs/Web/HTML>
- MDN Web Docs – CSS: <https://developer.mozilla.org/docs/Web/CSS>
- Walidator HTML: <https://validator.w3.org/>