

Podstawy informatyki

HTML5 / CSS3

Laboratorium 12

Semantyczne znaczniki tekstowe i dokumentacja techniczna

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

Cel ćwiczenia

Celem jest rozróżnienie semantyki tekstu od jego prezentacji oraz poprawne użycie elementów `strong`, `em`, `b`, `i`, `q`, `cite`, `abbr`, `dfn`, `code`, `pre`, `var`, `kbd`, `samp`, `mark`, `sup` i `sub`.

Wstęp

W pliku źródłowym ćwiczenie zostało omyłkowo opisane jako “Laboratorium L13”. W zestawie zachowano właściwy numer **L12**. Celem ćwiczenia jest rozróżnienie znaczników niosących znaczenie od czysto wizualnej stylizacji CSS.

Semantyka a wygląd

Nie należy upraszczać rozróżnienia do stwierdzenia, że `b` i `i` są zawsze “złe”. Współczesny HTML definiuje dla nich określone, słabsze znaczenia: `b` może oznaczać fragment zwracający uwagę bez dodatkowej ważności, a `i` treść o odmiennym jakości głosu/terminologii. Gdy jednak chodzi o ważność lub emfaticzny nacisk, właściwe są `strong` i `em`.

Ważność i emfaza

```
<p>
  Przed pomiarem <strong>odłącz zasilanie</strong>.
  Należy to zrobić <em>przed</em> podłączeniem miernika.
</p>
```

`strong` sygnalizuje istotność/ważność, a `em` nacisk, który może zmieniać sens wypowiedzi.

Cytaty, tytuły, skróty i definicje

```
<p>
  Autor napisał: <q>Najpierw należy wykonać pomiar.</q>
</p>
<p>
  Dane opisano w <cite>IEC 61508</cite>.
</p>
<p>
  Sterownik <abbr title="Programmable Logic Controller">PLC</abbr>
  steruje procesem.
</p>
<p>
  <dfn>Automatyka</dfn> zajmuje się sterowaniem procesami.
</p>
```

`cite` oznacza tytuł utworu/dzieła lub źródła, a nie nazwisko autora. Atrybut `title` w `abbr` może być pomocny, ale nie powinien być jedynym sposobem przekazania informacji niezbędnej do zrozumienia tekstu.

Kod, zmienne, klawiatura i wynik programu

```
<p>Ustaw <var>napiecieMax</var> na <code>230</code>.</p>
<p>Nacisnij <kbd>Ctrl</kbd> + <kbd>S</kbd>.</p>
<p>Program zwrocil: <samp>OK</samp>.</p>
```

`code` służy do fragmentów kodu, `var` do zmiennych/symboli, `kbd` do wejścia użytkownika, a `samp` do przykładowego wyjścia programu.

Blok kodu: `pre` + `code`

Sam `pre` opisuje tekst preformatowany i zachowuje białe znaki. Jeżeli jego zawartością jest kod, warto zagnieździć `code`:

```
<pre><code>for (let i = 0; i < 3; i++) {
  console.log(i);
}</code></pre>
```

Przykładowa stylizacja:

```
code {
  background: #f7f7f7;
  padding: 0.1em 0.3em;
  border-radius: 3px;
}

pre {
  background: #2c3e50;
  color: #ecf0f1;
  padding: 15px;
  border-left: 5px solid #3498db;
  overflow-x: auto;
}

pre code {
  background: transparent;
  padding: 0;
}
```

Dodatkowe znaczniki przydatne w dokumentacji

Znacznik	Znaczenie	Przykład
<code>mark</code>	treść wyróżniona w bieżącym kontekście	wynik wyszukiwania
<code>time</code>	data/czas w formie czytelnej maszynowo	<code>datetime="2026-09-17"</code>
<code>sup</code>	indeks górny	<code>m<sup>2</sup></code>
<code>sub</code>	indeks dolny	<code>H<sub>2</sub>O</code>
<code>samp</code>	przykładowe wyjście programu	<code><samp>ERROR 404</samp></code>

Przykład kompletny – fragment dokumentacji technicznej

```
<p>
  Przed uruchomieniem <strong>sprawdz polaryzacje zasilania</strong>.
</p>
```

```
W szczegolnosci <em>nie pomijaj</em> pomiaru napiecia.
</p>

<p>
  Sterownik <abbr title="Programmable Logic Controller">PLC</abbr>
  ustawia zmienna <var>predkosc</var> przez funkcje <code>setSpeed()</code>.
</p>

<p>Nacisnij <kbd>Enter</kbd>. Program powinien zwrocic <samp>READY</samp>.</p>

<pre><code>if (voltage > maxVoltage) {
  stopMotor();
}</code></pre>
```

HTML nadal przekazuje znaczenie po wyłączeniu CSS. To podstawowa różnica między semantyką a samym wyglądem.

Typowe błędy semantycznego formatowania

- używanie **strong** wyłącznie dlatego, że domyślnie jest pogrubiony,
- używanie **cite** do oznaczenia nazwiska autora zamiast tytułu dzieła,
- blok kodu w samym **pre** bez **code**, mimo że treść jest kodem,
- poleganie wyłącznie na tooltipie **abbr title** dla informacji kluczowej,
- mieszanie semantyki z dekoracją i dobieranie znaczników na podstawie wyglądu.

Przebieg ćwiczenia

Zadanie 1. Ważność i emfaza

Do wykonania

Wstaw akapit instrukcji bezpieczeństwa. Najważniejszy fragment oznacz **strong**, a fragment wymagający emfatycznego nacisku **em**. Ostyluj oba elementy w CSS, nie zmieniając ich znaczenia semantycznego.

Zadanie 2. Cytat, skrót i definicja

Do wykonania

Dodaj krótki cytat **q**, tytuł źródła **cite**, skrót PWM w **abbr title="Pulse Width Modulation"** oraz pojęcie definiowane w **dfn**. Upewnij się, że pełne rozwinięcie skrótu jest dostępne również w normalnym tekście, jeżeli jest istotne dla zrozumienia treści.

Zadanie 3. Dokumentacja kodu

Do wykonania

W jednym akapicie oznacz nazwę zmiennej przez **var**, fragment instrukcji programu przez **code**, klawisze przez **kbd** i przykładowy wynik programu przez **samp**.

Zadanie 4. Blok kodu

Do wykonania

Wstaw minimum pięcioliniowy fragment kodu w strukturze `<pre><code>...</code></pre>`. Ostyluj blok na wzór terminala i dodaj `overflow-x: auto`. Celowo dodaj jedną długą linię i sprawdź przewijanie.

Zadanie 5. Zapis techniczny

Do wykonania

Utwórz krótki opis wzoru lub jednostki wykorzystujący `sup/sub`, datę w `time` oraz jedno `mark`. Każdy znacznik powinien mieć rzeczywiste uzasadnienie semantyczne.

Zadanie 6. Analiza `b/i` kontra `strong/em`

Do wykonania

Przygotuj cztery krótkie przykłady: `b`, `i`, `strong` i `em`. Przy każdym dodaj komentarz HTML opisujący zamierzone znaczenie. Nie oceniaj elementu wyłącznie po domyślnym pogrubieniu lub kursywie.

Zadanie końcowe

Integracja

Przygotuj jedną stronę dokumentacji technicznej zawierającą ostrzeżenie, definicję, skrót, cytaty ze źródłem, kod liniowy, blok kodu, zmienną, wejście klawiaturowe i wynik programu. Stylizacja CSS ma poprawiać czytelność, ale struktura HTML musi pozostać zrozumiała także bez CSS.

Kryteria zaliczenia

- dobór znaczników według znaczenia, nie wyglądu,
- poprawne rozróżnienie `strong/em` od `b/i`,
- `pre+code` dla bloku kodu,
- użycie `samp` dla wyniku programu,
- rozwinięcie skrótu dostępne także bez polegania wyłącznie na dymku `title`.

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>
- Validator HTML: <https://validator.w3.org/>