Materiały pomocnicze:

W HTML5 znaczniki semantyczne służą do opisywania struktury i znaczenia treści na stronie w sposób bardziej czytelny dla przeglądarek, wyszukiwarek i programistów.

Lista najczęściej używanych znaczników semantycznych w HTML5:

1. `<header>` – **Definiuje nagłówek strony**, sekcji lub artykułu, zazwyczaj zawierający logo, menu nawigacyjne lub tytuły.
2. `<nav>` – **Określa sekcję nawigacyjną**, zawierającą linki do innych stron lub części dokumentu.
3. `<main>` – **Reprezentuje główną treść dokumentu**, unikalną dla danej strony (powinna występować tylko raz).
4. `<article>` – **Oznacza niezależną, samodzielną treść**, taką jak wpis na blogu, artykuł czy post.
5. `<section>` – **Grupuje powiązane tematycznie treści**, zwykle z nagłówkiem (np. `<h2>`).
6. `<aside>` – **Zawiera treści poboczne**, takie jak panele boczne, reklamy czy dodatkowe informacje.
7. `<footer>` – **Definiuje stopkę strony** lub sekcji, zawierającą np. informacje kontaktowe, prawa autorskie.
8. `<figure>` – **Służy do grupowania multimediów** (np. obrazów, diagramów) z opcjonalnym podpisem.
9. `<figcaption>` – **Podpis dla elementu** `<figure>`, **opisujący zawartość multimedialną**.
10. `<details>` – **Tworzy interaktywny element, który można rozwinąć/zwinąć**, aby pokazać dodatkowe informacje.
11. `<summary>` – **Definiuje nagłówek dla elementu** `<details>`, widoczny przed rozwinięciem.
12. `<mark>` – **Wyróżnia tekst**, który jest istotny w danym kontekście (np. wyniki wyszukiwania).
13. `<time>` – **Oznacza datę, godzinę lub zakres czasowy**, z opcjonalnym atrybutem `datetime`.
14. `<address>` – **Służy do oznaczania informacji kontaktowych**, np. adresu e-mail, telefonu czy lokalizacji.

15. <progress> – **Reprezentuje pasek postępu**, np. dla ładowania lub wypełnienia formularza.
16. <meter> – **Wskazuje wartość w określonym zakresie**, np. poziom naładowania baterii.
17. <dialog> – **Definiuje okno dialogowe lub modalne**, np. do wyświetlania alertów.
18. <picture> – **Umożliwia definiowanie różnych źródeł obrazów dla różnych urządzeń lub rozdzielczości**.
19. <template> – **Przechowuje treść, która nie jest wyświetlana od razu, ale może być użyta przez JavaScript**.

Znaczniki pomagają w lepszej organizacji kodu, poprawiają dostępność (accessibility) i optymalizację dla wyszukiwarek (SEO).

🔹 **1. Obiektowy sposób (OOP - Object-Oriented Programming, czyli programowanie obiektowe)**

```
$mysqli = new
mysqli("localhost","my_user","my_password","my_db");


if ($mysqli -> connect_errno) {
  echo "Błąd połączenia: " . $mysqli -> connect_error;
  exit();
}
```

🔹 **2. Proceduralny sposób**

```
$con =
mysqli_connect("localhost","my_user","my_password","my_db");

if (mysqli_connect_errno()) {
  echo "Błąd połączenia: " . mysqli_connect_error();
  exit();
}
```

🚀 Różnice

1. **Styl programowania**
    - OOP (new mysqli) – bardziej nowoczesny, wspiera podejście obiektowe, lepiej integruje się np. z frameworkami (Laravel, Symfony).
    - Proceduralny (mysqli_connect) – starszy styl, przypomina stary mysql_connect (już usunięty).
2. **Czytelność i rozszerzalność**
    - OOP daje możliwość używania metod ($mysqli->query(), $stmt->bind_param()), co sprawia, że kod jest bardziej spójny.

- Proceduralny miesza funkcje globalne z innymi elementami, więc w większych projektach kod może być mniej czytelny.

3. **Wydajność**
   - **Oba działają tak samo szybko** – pod spodem to ta sama biblioteka mysqli.

4. **Dobre praktyki**
   - Jeśli implementujesz **mały skrypt** (np. test, coś jednorazowego) – proceduralny jest szybszy do napisania.
   - Jeśli implementujesz **większą aplikację** – zdecydowanie lepiej trzymać się OOP (new mysqli), bo jest bardziej przejrzysty i łatwiej go łączyć z obiektowym stylem kodu.

✅ **Podsumowanie:**

- Oba sposoby są poprawne i tak samo szybkie.
- **Lepszy wybór**: obiektowy (new mysqli), bo jest nowocześniejszy, bardziej czytelny i łatwiej rozszerzalny w większych projektach.