



## Projekt praktyczny “Abstract Syntax Trees (AST)”

### Wprowadzenie:

Jesteś częścią zespołu odpowiedzialnego za nowy framework służący do budowania aplikacji webowych. Framework służący do budowania rozwiązań dla internetu musi być w stanie na bieżąco modyfikować i aktualizować strukturę strony, która w świecie internetu wyrażona jest w postaci kodu HTML.

Jako doświadczony twórca tego typu narzędzi wiesz już, że żaden programista nie zdecyduje się na pracę z surowym kodem HTML w postaci stringu, bo modyfikacja i aktualizowanie wybranych tagów i ich atrybutów byłaby bardzo uciążliwe. Zamiast pracy ze stringiem decydujesz się na pracę z tzw. Abstract Syntax Tree (AST), czyli obiektową reprezentacją danego fragmentu HTML.

Czym jest AST? Dla podanego fragmentu kodu HTML:

```
<div class="test">Hello world!</div>
```

AST w formacie JSON mógłby wyglądać tak:

```
{
  "ast": {
    "nodeType": "element",
    "tagName": "div",
    "attributes": [ { "name": "class", "value": "test" } ],
    "children": [
      {
        "nodeType": "text",
        "value": "Hello world!"
      }
    ]
  }
}
```

Reprezentacja kodu HTML jako AST pozwala programistom JavaScript na łatwiejsze modyfikowanie zawartości poszczególnych węzłów, dodawanie i usuwanie elementów oraz aktualizowanie atrybutów.

**Opis zadania i wymagania znajdziesz na kolejnej stronie.**

Dokument jest częścią kursu “Opanuj JavaScript Premium”.

<https://przeprogramowani.pl>



## Projekt praktyczny “Abstract Syntax Trees (AST)”

### Opis zadania:

W zespole, w którym pracujesz, trwa właśnie praca nad konwerterem HTML<->AST, aby umożliwić innemu zespołowi pracę na obiektowej reprezentacji kodu HTML zgodnie z założeniami framework'a. Jeden z członków zespołu napisał pierwszą część tego konwertera, która wyraża HTML w postaci AST, natomiast tobie przypisano zadanie polegające na odwróceniu tego procesu.

Stwórz moduł eksportujący funkcję **convertASTToString**. Jako parametr funkcja powinna przyjmować obiektową reprezentację fragmentu HTML (tzw. Abstract Syntax Tree). Wartością zwracaną powinien być fragment HTML w postaci stringu.

Przykładowy AST na którym możesz testować swoje rozwiązanie znajduje się tutaj:

<https://przeprogramowani.pl/examples/ast.json>

Möżesz zapisać powyższy plik jako .json i zimportować go do twojego projektu, lub skopiować jego zawartość i posługiwać się nim jak surowym obiektem, który jest parametrem wejściowym.

### Wymagania:

Po ukończeniu, funkcja ma spełniać następujące wymagania:

1. Wyjściem funkcji musi być poprawny kod HTML, który będzie możliwy do wyświetlenia przez przeglądarkę.
2. Struktura HTMLa powinna być dokładnym odwzorowaniem zawartości przekazywanego Abstract Syntax Tree - wszystkie elementy powinny zawierać zdefiniowane atrybuty i kolejne elementy potomne.

**UWAGA:** Zwracany HTML **nie musi** być poprawnie sformatowany a jego elementy **nie muszą** być zagnieżdżone. Głównym elementem zadania jest zwrócenie stringa z odpowiednim układem elementów względem siebie. Formatowanie możesz jednak potraktować jak wymaganie bonusowe.

Utknęłałeś? Sprawdź wskazówki na kolejnej stronie.



Fundamenty języka | Projekt Praktyczny: Abstract Syntax Trees

## Projekt praktyczny “Abstract Syntax Trees (AST)”

### Wskazówki:

1. Przed przystąpieniem do zadania zapoznaj się ze strukturą pojedynczego tagu HTML -  
<https://clearlydecoded.com/anatomy-of-html-tag>
2. Problem “zbudowania struktury HTML” powinieneś najpierw sprowadzić do zbudowania pojedynczego elementu z pojedynczego fragmentu AST. Elementy potomne możesz budować wywołując rekurencyjnie tę samą funkcję, ponieważ logika za każdym razem będzie taka sama.
3. Budowanie pojedynczego elementu to trzy kluczowe kroki - tag otwierający i atrybuty, elementy potomne, tag zamykający.

Dokument jest częścią kursu “Opanuj JavaScript Premium”.

<https://przeprogramowani.pl>