Politechnika Poznańska

Wydział Automatyki, Robotyki i Elektrotechniki

Instytut Robotyki i Inteligencji Maszynowej

Zakład Sterowania i Elektroniki Przemysłowej


# Aplikacją webowa HTML i JavaScript

# Aplikacje Mobilne i Wbudowane dla Internetu Rzeczy

Materiały do zajęć laboratoryjnych

dr inż. Dominik Łuczak, mgr inż. Adrian Wójcik

Dominik.Luczak@put.poznan.pl
Adrian.Wojcik@put.poznan.pl

# I. Cel zajęć

## Wiedzy

Celem zajęć jest zapoznanie z:

- znaczeniem i rolą języka HTML w aplikacjach webowych,
- składnią i podstawowymi tagami języka HTML,
- strukturą dokumentu HTML,
- znaczeniem i rolą obiektowego modelu dokumentu w aplikacjach webowych,
- znaczeniem i rolą języka JavaScript w aplikacjach webowych,
- podstawami składni języka JavaScript,

## Umiejętności

Celem zajęć jest nabycie umiejętności w zakresie:

- tworzenia dokumentów HTML,
- tworzenia skryptów JavaScript,
- komunikacji skryptów JS z dokumentami HTML,

## Kompetencji społecznych

Celem zajęć jest kształtowanie właściwych postaw:

- prawidłowego zarządzania bazą kodu aplikacji webowych,
- ugruntowanie rozumienia i świadomości znaczenia pozatechnicznych aspektów i skutków działalności inżyniera i związaną z tym odpowiedzialność za podejmowane decyzje,
- dobór właściwej technologii i narzędzi programistycznych do zadanego problemu,
- sprawdzenie prawidłowego działania przygotowanego systemu informatycznego.

# II. Polecenia końcowe

Wykonaj zadania laboratoryjne zgodnie z poleceniami i wskazówkami prowadzącego, pracując samodzielnie lub w dwuosobowym zespole. **Zachowaj zasady bezpieczeństwa podczas pracy!** Wykonaj raport laboratoryjny, dokumentujący prawidłowe wykonanie zadań. Wymogi redakcyjne oraz szablon sprawozdania dostępne są na plafromie *eKursy*. Raport oceniany jest w dwóch kategoriach: realizacja zadań i spełnienie wymogów redakcyjnych. Zrealizowane zadania są oceniane zero-jedynkowo. Spełnienie wymogów redakcyjnych oceniane jest procentowo. Sprawozdanie należy przesłać jako rozwiązanie zadania na platformie *eKursy* do niedzieli, 16 maja 2021 do 23:59.

# III. Przygotowanie do zajęć

## a) Zapoznanie z przepisami BHP

Wszystkie informacje dotyczące instrukcji BHP laboratorium są zamieszczone w sali laboratoryjnej oraz na stronie Zakładu [1]. Wszystkie nieścisłości należy wyjaśnić z prowadzącym laboratorium. Wymagane jest zaznajomienie i zastosowanie do regulaminu.

Na zajęcia należy przyjść przygotowanym zgodnie z tematem zajęć. Obowiązuje również materiał ze wszystkich odbytych zajęć.

## B) Wprowadzenie do HTML

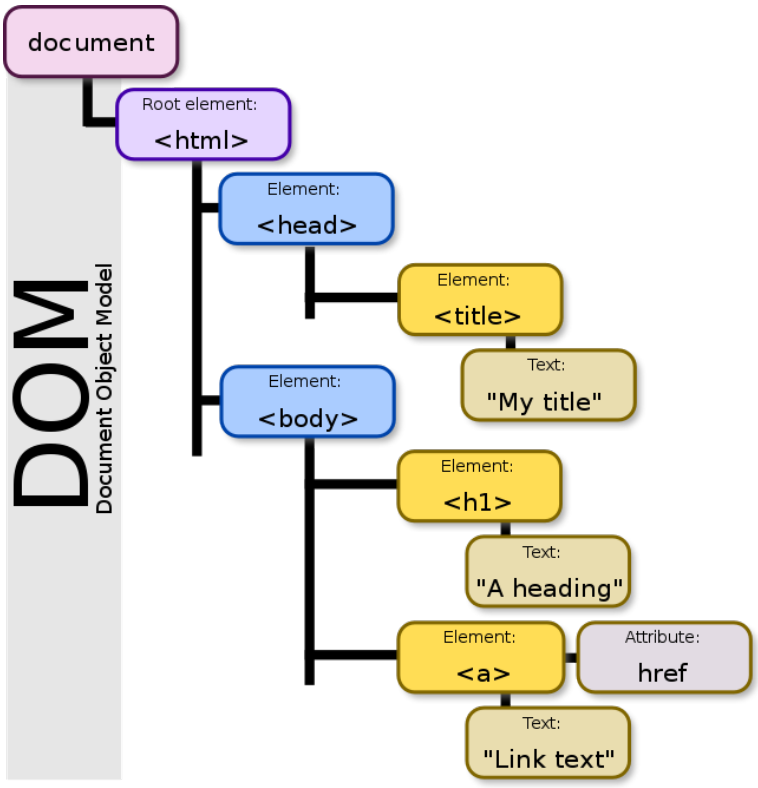
HTML (ang. *HyperText Markup Language*) czyli hipertekstowy język znaczników stanowi najbardziej podstawowy elementem składowym sieci World Wide Web. Definiuje **strukturę** treści stron internetowych. Hipertekst (ang. *hypertext*) odnosi się do linków łączących ze sobą strony internetowe, zarówno w obrębie jednej strony internetowej, jak i pomiędzy różnymi stronami internetowymi. Element HTML jest odróżniany od zwykłego tekstu w dokumencie za pomocą znaczników (ang. *markup*) lub „tagów", które składają się z nazwy elementu otoczonego przez „<" oraz „>". Nazwa elementu wewnątrz znacznika nie rozróżnia wielkości liter. Oznacza to, że może być napisana wielkimi, małymi lub zmieszanymi literami. Na przykład tag `<title>` może być zapisany jako `<Title>` , `<TITLE>` lub w inny sposób [2].

Aktualnie (od maja 2019 roku) organizacja WHATWG (ang. *Web Hypertext Application Technology Working Group*) tj. Grupa Robocza ds. Technologii Hipertekstowych Aplikacji Sieciowych jest jedynym wydawcą standardów HTML i DOM. Standard języka znaleźć można na stronie WHATWG [3]. Aktualną wersją języka jest HTML5 [4]. Listę tagów wraz z opisem i przykładami znaleźć można na portalu devdocs.io [5].

Osobnymi technologiami poza HTML stosowanymi do tworzenia aplikacji webowych są: **CSS** używany do opisu wyglądu/prezentacji strony internetowej oraz **JavaScript** używany do opisu funkcjonalności/zachowania aplikacji [2].

## C) Obiektowy model dokumentu

DOM (ang. *Document Object Model*) tj. Obiektowy Model Dokumentu to API, które reprezentuje dokumenty HTML i pozwala na programowe komunikowanie się z nimi. DOM jest modelem dokumentu prezentowanego w przeglądarce reprezentowanego jako drzewo, w którym każdy węzeł reprezentuje część dokumentu (patrz: rys. 1).




*Rys. 1. Przykład hierarchii DOM w dokumencie HTML [6].*

Wprowadzenie strukturalnego odwzorowania dokumentu, pozwala modyfikować jego zawartość i warstwę prezentacji. DOM jest jednym z najczęściej używanych API w WWW ponieważ pozwala ono na dostęp i złożoną interakcję z tzw. *węzłami* (ang. *nodes*) dokumentu. Węzły te mogą być tworzone, przenoszone i zmieniane. Do węzłów można także dodać nasłuchiwanie zdarzeń (ang. event listener). DOM łączy więc strony WWW ze skryptami JavaScript (lub innymi językami programowania). Standard DOM znaleźć można na stronie WHATWG [7].

### d) Wprowadzenie do JavaScript

JavaScript (w skrócie: *JS*) to skryptowy - interpretowany lub kompilowany metodą JIT (ang. just-in-time) - język programowania. Charakterystyczną cechą JS jest to że, funkcje są obywatelami pierwszej kategorii", tj. obiektami, które można przechowywać w zmiennych jako referencje i przekazywać jak każde inne obiekty. JavaScript jest językiem wieloparadygmatowym: opartym na prototypach, dynamicznej składni, zorientowanym obiektowo, imperatywnym i deklaratywnym (programowanie funkcyjne) [8].

Standardem dla JavaScript jest ECMAScript [9]. W czerwcu 2015 roku, organizacja ECMA International opublikował szóstą główną wersję ECMAScript. Od roku 2012, wszystkie nowoczesne przeglądarki całkowicie obsługują ECMAScript 5.1. Starsze przegladarki obsługują co najmniej ECMAScript 3. Aktualnie standardy ECMAScript są wydawane w cyklach rocznych.

Nie należy mylić JavaScript z językiem programowania Java. Zarówno „Java" jak i „JavaScript" są znakami towarowymi przedsiębiorstwa Oracle. Jednak obydwa te języki programowania mają bardzo różną składnię, semantykę i zastosowanie. Nazwa ta jest przede wszystkim wynikiem zabiegów marketingowych.

JavaScript jest najbardziej znany jako język skryptowy dla stron internetowych, używa go jednak również wiele środowisk poza przeglądarką, takich jak Node.js, Apache CouchDB czy Adobe Acrobat.

#### 1. JavaScript w dokumencie HTML

Na listingu 1. zaprezentowano przykład dokumentu HTML ze skryptem napisanym w języku JavaScript, pozwalającym na obsługę zdarzenia w postaci naciśnięcia przycisku. Całość znajduje się w pojedynczym pliku **.htm**.

*Listing 1. Przykład dokumentu HTML ze skryptem JavaScript w jednym pliku.*

```
01. <!DOCTYPE html>
02. <html>
03.   <!-- HTML file head: web page metadata -->
04.   <head>
05.     <title>HTML/JavaScript examples</title>
06.     <script>
07.       var buttonCounter = 0;
08.       function myOnClickMethod() {
09.         buttonCounter += 1;
10.         var paragraphDispText ="<b>Click␣counter:␣</b>"+buttonCounter.toString();
11.         document.getElementById("paragraph").innerHTML = paragraphDispText;
12.       }
13.     </script>
14.   </head>
15.   <!-- HTML file body: web page content -->
16.   <body>
17.     <h1>Simple button example: signle file</h1>
18.       <button onclick="myOnClickMethod()">Click me</button>
19.       <p id="paragraph"></p>
20.   </body>
21. </html>
```

Tworząc aplikacje webowe dobrą praktyką jest jednak separacja opisu *struktury dokumentu* (HTML) od *programowej funkcjonalności* (JavaScript). Na listingach 2. i 3. przedstawiono odpowiednio plik **.htm** (zawierający informację o wykorzystanym pliku **.js**) oraz plik **.js**. Oba przedstawione aplikacje działają oczywiście w identyczny sposób.

*Listing 2. Przykład dokumentu HTML ze skryptem JavaScript w osobnym pliku.*

```
<!DOCTYPE html>
<html>
  <!-- HTML file head: web page metadata -->
  <head>
    <title>HTML/JavaScript examples</title>
    <script type="text/javascript" src="scripts/button_example_v2.js"></script>
  </head>
  <!-- HTML file body: web page content -->
  <body>
    <h1>Simple button example: separte HTML and JavaScript</h1>
      <button onclick="myOnClickMethod()">Click me</button>
      <p id="paragraph"></p>
  </body>
</html>
```

*Listing 3. Przykład skryptu JavaScript w osobnym pliku.*

```
var buttonCounter = 0; ///< button onClick event counter

/* @brief button onClick event handling */
function myOnClickMethod() {
  buttonCounter += 1;
  var paragraphDispText = "<b>Click counter: </b>" + buttonCounter.toString();
  document.getElementById("paragraph").innerHTML = paragraphDispText;
}
```

Za pomocą tagu `<script>` można dodawać do dokumentu zewnętrzne biblioteki JS dostępne zarówno lokalnie jak i jako zasób internetowy.

## 2. JSON w języku JavaScript

Format wymiany danych JSON (JavaScript Object Notation) wywodzi się bezpośrednio z języka JavaScript - obiekty JSON stanowią **tekstową** reprezentację obiektów w języku JavaScript. Funkcję pozwalające na parsowanie, tj. uzyskanie obiektu JS z teksu w formacie JSON (`JSON.parse()`) oraz kodowanie, tj. uzyskanie tekstu w formacie JSON z obiektu JS (`JSON.stringify()`) stanowią część języka JS i nie wymagają zastosowania dodatkowych bibliotek. Na listingu 4. przedstawiono przykład zastosowania tych funkcji.

*Listing 4. Przykład wykorzystania funkcji `JSON.parse()` i `JSON.stringify()`.*

```
const jsObjExample = {
    temperature: { value: 20.3,   unit: "C"   },
    pressure:    { value: 1023.0, unit: "hPa" },
    humidity:    { value: 63,     unit: "%"   }
};

const jsonTextExample ='{"temperatur":{"value":20.3,"unit":"C"}, "pressure":{"
    value":1023.0,"unit":"hPa"},"humidity":{"value":63,"unit":"%"}}';
var jsObj = JSON.parse(jsonTextExample);
var jsonText = JSON.stringify(jsObjExample);
```

### 3. Debugowanie JS w przeglądarce

Większość współczesnych przeglądarek umożliwia wykorzystanie *konsoli WWW* do podglądu zmiennych oraz egzekucję kodu języka JavaScript w aktualnym *kontekście*. Dostępne są również narzędzia pozwalające na pełne debugowanie aplikacji webowej wbudowane w przeglądarce - np. Firefox czy Chrome. Na rys. 2 przedstawiono przykład debugowania prostej aplikacji do obsługi przycisku.


*Rys. 2. Przykład debugowania skryptu JS w przeglądarce Chrome - zatrzymanie egzekucji skryptu za pomocą breakpointa w linijce 8. po naciśnięciu przycisku.*

## IV. Scenariusz do zajęć

### a) Środki dydaktyczne

| | |
|---|---|
| Sprzętowe | • komputer PC, |
| Programowe | • przeglądarka internetowa (np. Firefox, Chrome), |
| | • edytor tekstu (np. Notepad++, VS Code), |
| | • serwer WWW (np. XAMPP) (*opcjonalnie*), |

### b) Zadania do realizacji

1. Stwórz prosty dokument HTML z pojedynczym przyciskiem.
   (a) Za pomocą edytora tekstu stwórz dokument HTML (*.htm) oraz skrypt JavaScript (*.js).
   (b) W dokumencie HTML umieć sekcję nagłówkową oraz ciało strony. Nadaj dokumentowi tytuł, dodaj informację o skrypcie oraz umieść pojedynczy przycisk o treści „CLICK ME”. Przypisz atrybutowi `onClick` funkcję, którą następnie zdefiniujesz w skrypcie JavaScript.
   (c) W skrypcie JavaScript zdefiniuj funkcję do obsługi przycisku. Funkcja powinna zmieniać treść przycisku na „HELLO WORLD” oraz inkrementować globalną zmienną.
   (d) Otwórz dokument HTML za pomocą przeglądarki i sprawdź, czy zdefiniowana funkcjonalność działa prawidłowo.
   (e) Otwórz konsolę WWW przeglądarki i sprawdź czy zmienna globalna zdefiniowana w skrypcie prawidłowo zmienia wartość po każdorazowym kliknięciu przycisku.
   (f) Alternatywnie umieść dokument i skrypt na lokalnym serwerze, a następnie otwórz dokument wpisując odpowiedni URL w przeglądarce.

2. Stwórz prostą aplikację webową: konwerter jednostek.
   (a) Stwórz dokument HTML zawierający elementy, pozwalające na wprowadzenie przez użytkownika wartości liczbowej (np. prędkości obrotowej), wybór jednostki wejściowej i wyjściowej (min. 4, np. rpm rps, rad/s, rad/min, deg/s, deg/min), zatwierdzenie danych wejściowych oraz prezentację wyniku. Wykorzystaj tagi `input`, `select` i `output`.
   (b) Stwórz skrypt JavaScript, umożliwiający dokonywanie konwersji jednostek. Spróbuj zaimplementować tę funkcjonalność nie wykorzystując instrukcji warunkowych.
3. Zmodyfikuj prostą aplikację webową wykorzystując JSON jako wejście i wyjście.
   (a) Bazując na rozwiązaniu poprzedniego zadania, stwórz aplikację webową, która dokonuję konwersji jednostek (np. prędkości obrotowej) na podstawie podanej przez użytkownika struktury w formacie JSON. Struktura musi zawierać informację o wielkości wejściowej, jednostce wejściowej oraz jednostce wyjściowej. Zamieść przykładową strukturę jako wartość początkową pola tekstowego.
   (b) Wyjście aplikacji również należy zaprezentować jako tekst w formacie JSON, zawierający informację o wartości wyjściowej oraz jednostce wyjściowej.
   (c) Dokonaj walidacji danych wejściowych, poinformuj użytkownika o ewentualnych błędach (np. struktura niezgodna z formatem JSON, błędna wartość wejściowa, błędna/niedostępna jednostka wejściowa, błędna/niedostępna jednostka wyjściowa).
4. Stwórz aplikację webową zawierająca wykres funkcji z wykorzystaniem zewnętrznej biblioteki dla JavaScript.
   (a) Zapoznaj się z dokumentacją biblioteki Chart.js dostępnej na stronie chartjs.org.
   (b) Stwórz dokument HTML i dodaj do niego bibliotekę Chart.js za pomocą:
   ```
   <script type="text/javascript"
       src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0">
   <\script>
   ```
   Dodaj również własny skrypt. Kolejność umieszczania skryptów w dokumencie jest istotna w kontekście widoczności deklarowanych w skryptach zmiennych.
   (c) W dokumencie umieść wykres liniowy (*line chart*) - bazując na przykładach w dokumentacji biblioteki Chart.js. Dodaj pola pozwalające na wprowadzenie przez użytkownika danych. Dokument powinien umożliwiać wprowadzenie informacji nt. amplitudy, fazy i okresu sygnału harmonicznego oraz okresu próbkowania.
   (d) W skrypcie zaimplementuj funkcjonalność, pozwalająca na wygenerowanie danych do wykresu na podstawie parametrów sygnału harmonicznego przekazanych przez użytkownika oraz aktualizację danych oraz opis osi x wykresu.
5. Stwórz szkielet aplikacji webowej składającej się z wielu dokumentów HTML.
   (a) Stwórz dokument HTML składający się z trzech elementów: nagłówek (tytuł rozdziału), 4 hiperłącza oraz ramkę.
   (b) Hiperłącza powinny stanowić odsyłacze do rozwiązać poprzednich zadań. Po kliknięciu hiperłącza, ramka powinna zostać wypełniona treścią odpowiedniego dokumentu.
   (c) Odpowiedz na pytanie: Czy JavaScript jest niezbędny do osiągnięcia tej funkcjonalności?

## Bibliografia

1. *Regulaminy porządkowe i instrukcje BHP* [online]. [B.d.] [udostępniono 2019-09-30]. Dostępne z: http://zsep.cie.put.poznan.pl/materialy-dydaktyczne/MD/Regulaminy-porz%C4%85dkowe-instrukcje-BHP/.

2. *HTML: Hypertext Markup Language* [online]. [B.d.] [udostępniono 2020-04-14]. Dostępne z: https://developer.mozilla.org/en-US/docs/Web/HTML. Library Catalog: developer.mozilla.org.
3. *HTML Standard* [online]. [B.d.] [udostępniono 2020-04-14]. Dostępne z: https://html.spec.whatwg.org/multipage/.
4. *HTML5* [online]. [B.d.] [udostępniono 2020-04-14]. Dostępne z: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5. Library Catalog: developer.mozilla.org.
5. *HTML documentation — DevDocs* [online]. [B.d.] [udostępniono 2020-04-14]. Dostępne z: https://devdocs.io/html/.
6. *Document Object Model* [online]. 2019 [udostępniono 2020-04-14]. Dostępne z: https://en.wikipedia.org/w/index.php?title=Document_Object_Model&oldid=932262242. Page Version ID: 932262242.
7. *DOM Standard* [online]. [B.d.] [udostępniono 2020-04-14]. Dostępne z: https://dom.spec.whatwg.org/.
8. *JavaScript* [online]. [B.d.] [udostępniono 2020-04-14]. Dostępne z: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Library Catalog: developer.mozilla.org.
9. *Standard ECMA-262* [online]. [B.d.] [udostępniono 2020-04-14]. Dostępne z: https://www.ecma-international.org/publications/standards/Ecma-262.htm.