JavaScript - DOM

Autor: Marcin Dłubis

1 Spis treści

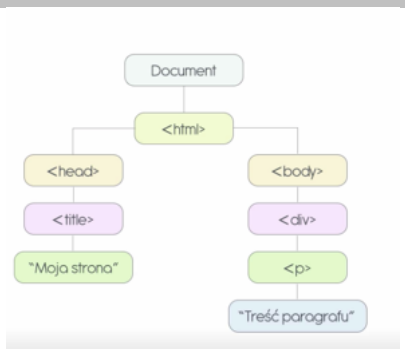
Manipulowanie DOMem

1.1 Czym jest DOM?

DOM - Obiektowy Model Dokumentu (ang. Document Object Model)

Taki kod zostaje przesłany z serwera do przeglądarki. Przeglądarka parsuje ten tekst na DOM.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Moja strona</title>
</head>
<body>
    <div> <p> Treść </p> </div>
</body>
</html>
```



W takiej formie dobrze widać, że np. element jest rodzicem elementu <p>,ale jest dzieckiem elementu <body>

1.2 Typy węzłów (node types)

- Document
- Element (div, p)
- Attribute (do elementów HTML, jak np. width itp)
- Text
- Comment

Przejdź do dowolnej strony, np.

https://pl.wikipedia.org/wiki/Wikipedia:Strona_g%C5%82%C3%B3wna

To, co widzisz, to już wyrenderowana strona!

Następnie przejdź do kodu strony (w Chrome jest to Widok -> Programista -> Wyświetl Źródło).

Jest to nic innego jak tekst, który przeglądarka zamienia na obiektowy model dokumentu.
Tekst ten został przysłany z serwera. Jeżeli znajdują się tam odwołania do jakichś skryptów, ten skrypt się wykona i stworzy nowy element i go doda, to w źródle strony tego nie zobaczymy.

Aby zobaczyć zmiany, trzeba przejść do PPM -> Zbadaj element.

ZADANIE:

usuń jakiś element kodu HTML z inspektora i zobacz, co się stanie.

1.3 DOM API vs. JavaScript

```
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Document</title>
</head>
<body>
   <div> Treść </div>
</body>
</html>
```

Możemy np. wyszukać taki element poprzez.

```
var diva = document.querySelector(”div”);
```

Dzięki temu możemy:

znaleźć dzieci tego elementu, dodać atrybut, usunąć klasę, element czy dodać zdarzenie.
Pamiętaj: DOM to nie JavaScript!

Przykład:

wykonując kod JS np. alert(„Witaj świecie”) pokaże nam się okienko z komunikatem. Ale samo okienko z komunikatem to już nie jest JS, tylko to, co jest wbudowane w przeglądarkę!
DOM API jest to zestaw interfejsów programistycznych, które pozwalają nam komunikować się z przeglądarką (czyli mamy obiekty, metody itp.)
Przykłady API:

DOM API - umożliwia manipulowanie drzewem DOM
XMLHttpRequest API - umożliwia wysłanie żądania na serwer
Audio & Video API - umożliwia odtwarzanie multimediów na stronie bez pluginów (czyli bez flasha, silverlight)
Canvas API - umożliwia rysowanie
File API - umozliwia czytanie treści wybranych plików
Geolocation API - umożliwia pobranie współrzędnych GPS urządzenia

2 Znajdowanie elementów DOM

Pamiętaj - kiedy za pomocą JavaScriptu będziesz dodawać, usuwać lub zmieniać jakieś elementy strony, będą to operacje na DOM, nie na HTML.
Aby zmodyfikować element DOM za pomocą JS, musimy go wskazać, co umożliwią nam opisane poniżej funkcje.

getElementById

Ta funkcja umożliwia wskazanie elementu o konkretnym identyfikatorze (id).

Składnia:

```
var contentBox = document.getElementById(‘example');
```

Powyższy kod wybierze z DOM element o id ‘example’ i zapisze go w stałej.

querySelector

Ta funkcja zwróci pierwszy element w dokumencie o danej klasie.

Składnia:

```
var mainHeader = document.querySelector('main-header');
```

Powyższy kod wybierze z DOM element o klasie main-header i zapisze go w stałej.

querySelectorAll

Ta funkcja wyszuka wszystkie elementy, pasujące do selektora.

Składnia:

```
var links = document.querySelectorAll('.paragraphs a');
```

Powyższy kod wybierze z DOM wszystkie elementy pasujące do selektora .paragraphes a i zapisze je w stałej.

Przykłady:

```
<body>
<h1> Przykład </h1>
<div id=”container”>
<p class=”text”> Przykładowy tekst </p>
<p class=”text”> Przykładowy tekst </p>
<a href=””> Link </a>
</div>
```

chcemy np. pobrać zmienną container

Do pobierania ID służy metoda getElementByID

```
var links = document.querySelectorAll('.paragraphs a');
```

Do pobierania elementów typu np. p służy metoda getElementByTagName

```
var pobranie_elementu = document.getElementByTagName(”p”);
```

Do pobierania klasy służy metoda getElementsByClassName

```
var pobranie_klasy = document.getElementsByClassName(”text”);
```

Do pobrania tylko jednego elementu, ID czy klasy możemy wykorzystać:

```
var pobranie_querySelector = document.querySelector("h1");
var pobranie_querySelector1 = document.querySelector(".text");
var pobranie_querySelector2 = document.querySelector("#container");
```

Po pobrania wszystkich elementów:

```
var pobranie_nowe = document.querySelectorAll(”a”);
```

3 Manipulacja elementami DOM

3.1 classList

Parametr classList zawiera zestaw metod, które pozwalają na operacje na klasach danego elementu.

Metoda	Zastosowanie
add	dodanie klasy
remove	usunięcie klasy
toggle	Przełączanie pomiędzy klasami (dodanie klasy jeśli jej nie było lub usunięcie, jeżeli była)
contains	sprawdzenie czy element zawiera daną klasę

Przykłady:

```
var div = document.querySelector('#testing-div');
```

```
div.classList.add('active');
console.log(div.classList.contains('active')); // true
```

```
div.classList.remove('active');
console.log(div.classList.contains('active')); // false
```

```
div.classList.toggle('active');
console.log(div.classList.contains('active'));
```

3.2 innerHTML

```
var div = document.querySelector('#testing-div');

console.log(div.innerHTML);

div.innerHTML = '<strong>Hello world!</strong>'
```

3.3 insertAdjacentHTML

Metoda insertAdjacentHTML służy do wstawiania kodu HTML bez usuwania zawartości danego elementu. Jako pierwszy argument podaje się tekst, który decyduje o miejscu wstawienia kodu HTML.

Argument	Miejsce wstawienia kodu
``'beforebegin'	przed elementem
'afterbegin'	w elemencie, na początku jego zawartości
'beforeend'	w elemencie, na końcu jego zawartości
'afterend'	po elemencie

Składnia:

```
var div = document.querySelector('#testing-div');
var newCode = '<strong>Hello world!</strong>';

div.insertAdjacentHTML('beforeend', newCode);
```

3.4 insertAdjacentElement

Ta metoda działa analogicznie do metody insertAdjacentHTML. Różni się tylko tym, że drugim argumentem będzie element DOM, a nie kod HTML.

Składnia:

```
var div = document.querySelector('#testing-div');

var newLink = document.createElement('a');
newLink.setAttribute('href', 'https://wikipedia.com');
newLink.innerHTML = "Wikipedia";

div.insertAdjacentHTML('beforeend', newLink);
```

3.5 createElement

Tworzenie nowego elementu DOM można osiągnąć za pomocą metody createElement wykonanej na obiekcie document. Element zostanie jedynie stworzony i zwrócony – nie zostanie dodany na stronie.

```
Var newDiv = document.createElement('div');
```

3.6 Remove

Usunięcie elementu z DOM można wykonać za pomocą metody remove wykonanej na elemencie, który ma zostać usunięty.

```
var galleryDiv = document.querySelector('div.gallery');
galleryDiv.remove();
```

4 Zdarzenia

```
<button onclick="changeText()">Kliknj mnie! </button>
<div id="sample"></div>

<script>
function changeText() {
  document.getElementById("sample").innerHTML = "działa?";
}
</script>
```

Lub

```
<button id="sampleBtn">Kliknj mnie! </button>
<div id="sample"></div>

<script>
document.getElementById("sampleBtn").onclick = changeText;

function changeText() {
  document.getElementById("sample").innerHTML = "działa?";
}</script>
```

```
<button id="sampleBtn">Kliknj mnie! </button>
<div id="sample"></div>

<script>
document.getElementById("sampleBtn").addEventListener("click", changeText);

function changeText() {
  document.getElementById("sample").innerHTML = "działa?";
}}</script>
```

```
<button id="sampleBtn">Kliknj mnie! </button>
<div id="sample"></div>

<script>
document.getElementById("sampleBtn").addEventListener("click", function() {
   document.getElementById("sample").innerHTML = "działa?";
});
</script>
```

Zadanie domowe:

Przećwicz nie tylko ze zdarzeniem ‘onlick’, ale sprawdź jak działają:

- Onload
- Onfocus
- Onmousedown
- Onmouseup
- Onmouseover
- Onmouseout

5 Dobre praktyki

Lista kilku dobrych praktyk, o których warto pamiętać podczas pisania skryptów.

Nazwy

Dla nazw zmiennych i funkcji:

- używaj formatu *camelCase* (bez spacji pomiędzy poszczególnymi słowami, każde kolejne słowo zaczynając wielką literą),
- używaj tylko języka angielskiego,
- używaj tylko znaków alfanumerycznych (cyfr i liter, ale bez polskich znaków).

Spacje

Wstawiaj spację przed i za operatorami (czyli znakami =, +, -, *, /) oraz po przecinkach.

Średniki

Wstawiaj średnik na końcu każdej linii kodu, z wyjątkiem:

- deklaracji funkcji nazwanych, np. function myFunc(){},
- bloków if, else if i else,
- pętli.

Wcięcia

Używaj dwóch spacji, aby tworzyć wcięcia w blokach kodu.

Strict mode

Dobrą praktyką jest uruchamianie kodu JS w "trybie ścisłym" poprzez umieszczenie na początku każdego pliku ze skryptami następującej linii:

```
'use strict';
```

Dzięki tej deklaracji pomyłki, które normalnie nie wywołałyby błędu, będą traktowane jak błąd i wyświetlane na czerwono w konsoli.