infoShare ACADEMY

JavaScript osadzanie skryptów na stronie i DOM API

infoShare Academy

infoShareAcademy.com

HELLO

Maciej Mikulski

Senior Front End Developer
@JIT.Team @Dialecticanet.com

infoShare ACADEMY

01. Osadzanie skryptów

infoShareAcademy.com

Script tag

Użycie bezpośrednio:

```
<script>
  console.log('boom!');
</script>
```

Wczytanie kodu JavaScript z pliku:

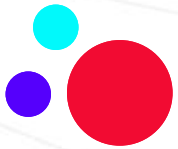
```
<script src="script.js"></script>
```

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script

Defer, async

Kolejność wykonywania skryptów

- Skrypty wykonywane są w kolejności znalezionej w HTML w sposób blokujący, tzn. że dopóki skrypt nie zostanie wczytany i wykonany dalsza część html nie jest przetwarzana
- Wyjątkiem są tagi skrypt z atrybutami async i defer
  - async – wczytwyanie skrypt zostanie rozpoczęte bez blokowania, a wykonanie nastąpi po jego wczytaniu; wykonanie skryptów wg zasafy kto pierwszy ten lepszy
  - defer – rozpocznie ładowanie skryptu, ale opóźni uruchomienie do momentu przed wywołaniem eventu DOMContentLoaded; wykonanie skryptów wg kolejności w dokumencie



Klasyczna sztuczka

Wykonaj kod po evencie DOMContentLoaded, czyli po wczytaniu całego dokumentu HTML.

```
<script>
        document.addEventListener("DOMContentLoaded", function(){
            console.log('Hello after DOMContentLoaded!');
        });
</script>
```

Browser cache

Aby przyspieszyć ładowanie strony przeglądarki zapamiętują zawartość pobranych plików, tak aby uniknąć ich ponownego ładowania.
Aby wymusić ponowne pobranie możemy zmodyfikować nazwę pliku lub parametry:

```
<script src="script.js?version=12kj3h21j3" />
<script src="script.12kj3h21j3.js" />
```

02. DOM API

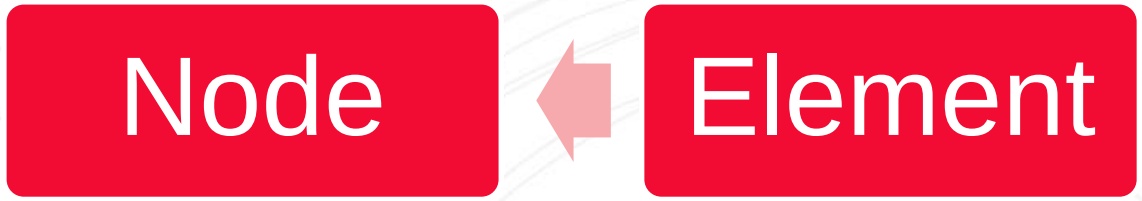
Document Object Model

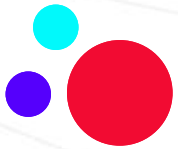
infoShareAcademy.com

infoShare ACADEMY

Do poczytania

- DOM: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
- document: https://developer.mozilla.org/en-US/docs/Web/API/Document
- window: https://developer.mozilla.org/en-US/docs/Web/API/Window
- element: https://developer.mozilla.org/en-US/docs/Web/API/Element
- node: https://developer.mozilla.org/en-US/docs/Web/API/Node





Wyszukiwanie elementów DOM

Zwracają pierwszy element spełniający kryterium wyszukiwania

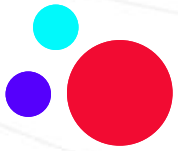
- document.querySelector('div.avatar')
- document.getElementById('1234')

Zwraca listę Node'ów (NodeList)

- document.querySelectorAll('div.avatar')

Zwraca HTMLCollection

- document. getElementsByTagName('p')
- document. getElementsByClassName('.avatar')



Powiązane elementy DOM

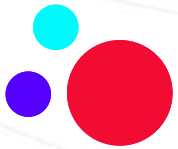
node.childNodes

element.children

node.firstChild

node.lastChild

node.parentElement



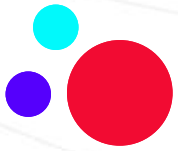
Tworzenie elementów DOM

- Tworzenie elementów i dodawanie atrybutów

```
const oliwa = document.createElement('li');
oliwa.textContent = 'oliwa’;
```

- Dodawanie element do istniejących elemntów

```
const lista = document.querySelector('#caprese ul');
lista.appendChild(oliwa);
```



Modyfikacja atrybutów elementów

```
const element = document.querySelector('li')
element.textContent = 'nowy tekst'
```

Porównaj atrybuty textContent i innerHTML

Dodawanie atrybutów

```
element.setAttribute('src', 'index.js')
```

Usuwanie atrybutów

```
element.removeAttribute('src')
```

Sprawdzanie czy atrybut istnieje

```
element.hasAttribute('src')
```

Atrybut classList

```
const element = document.querySelector('li')
```

Wypisz klasy:

```
element.classList
```

Dodaj klasę:

```
element.classList.add('mojaKlasa')
```

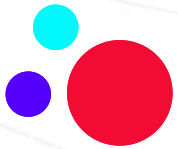
Usuń klasę:

```
element.classList.remove('mojaKlasa')
```

Zamień klasę:

```
element.classList.replace('staraKlasa',  'nowaKlasa')
```

Atrybut `className` zwraca wszystkie klasy w postaci jednego ciagu znaków – tak jak widzielibyśmy je w zapisie html.



Style: computed style vs style

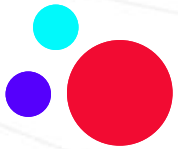
```
const element = document.getElementsByClassName('element')[0]
```

Metoda style na elemencie zwróci nam style bezpośrednio ustawione na elemencie za pomocą metody style lub inline css

```
element.style.color
```

Metoda window.getComputedStyle(element) pozwala na dostęp do efektywnych styli, uwzględniając wszystkie style wpływające na element (z plików css, style przeglądarki itd.)

```
window.getComputedStyle(element).color
```



Przenoszenie elementów

Wstaw element po elemencie:

- Element.after()

Wstaw element przed elementem:

- Element.before()

Wstaw element jako pierwsze dziecko:

- Element.prepend()

Wstaw element jako ostatnie dziecko:

- Element.append()

Usuń element:

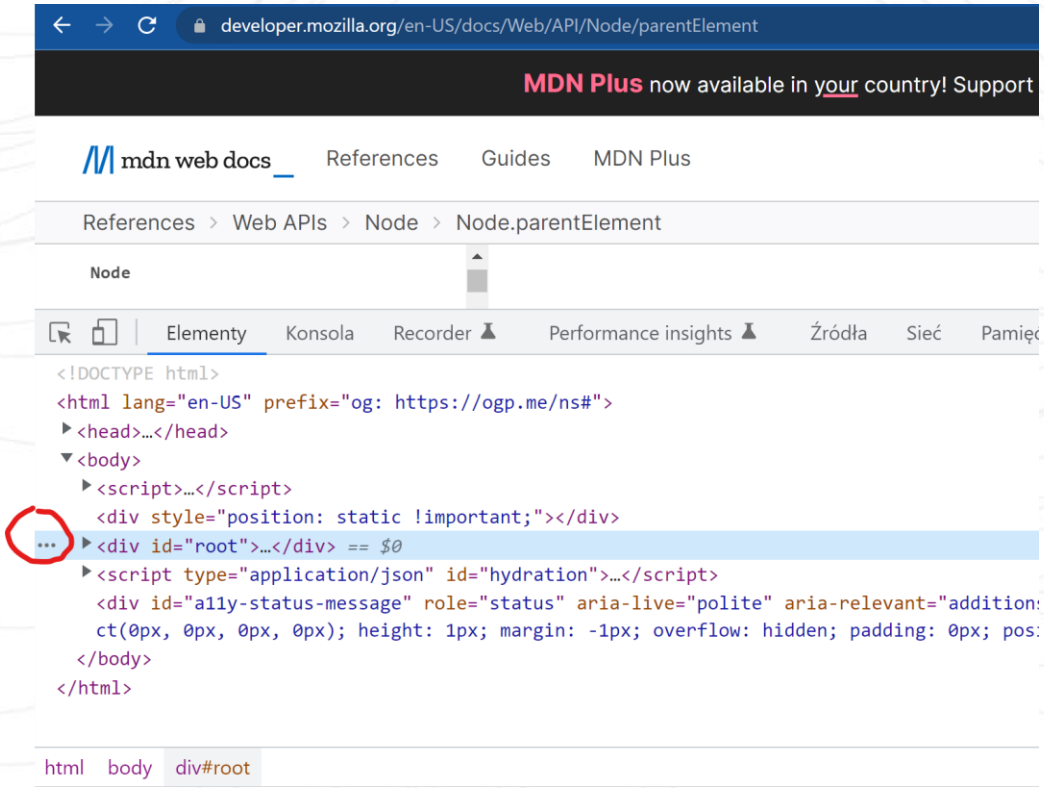
- Element.remove()

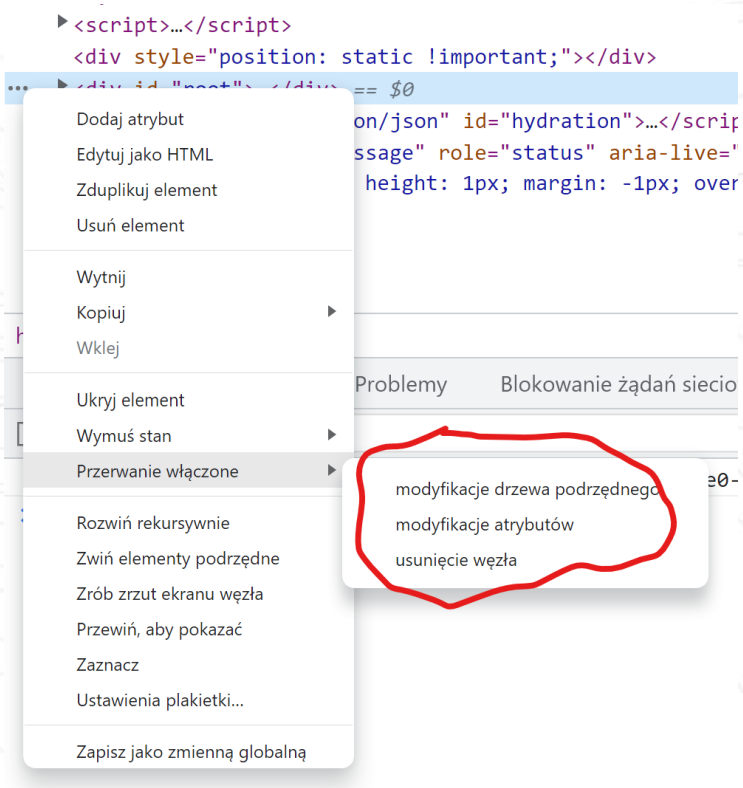
Zamień na:

- Element.replaceWith()

Śledzenie zmian za pomocą debuggera

Debugger przeglądarki pozwala na ustawienie breakpointów na modyfikacje atrybutów element, zmian w jego drzewie oraz usunięcie element.





Dziękuję za uwagę!

infoShareAcademy.com