JavaScript - podstawy - DOM

v3.0.0

Plan

- Wprowadzenie
- Wyszukiwanie elementów w DOM-ie
- Więcej o elemencie

Wprowadzenie

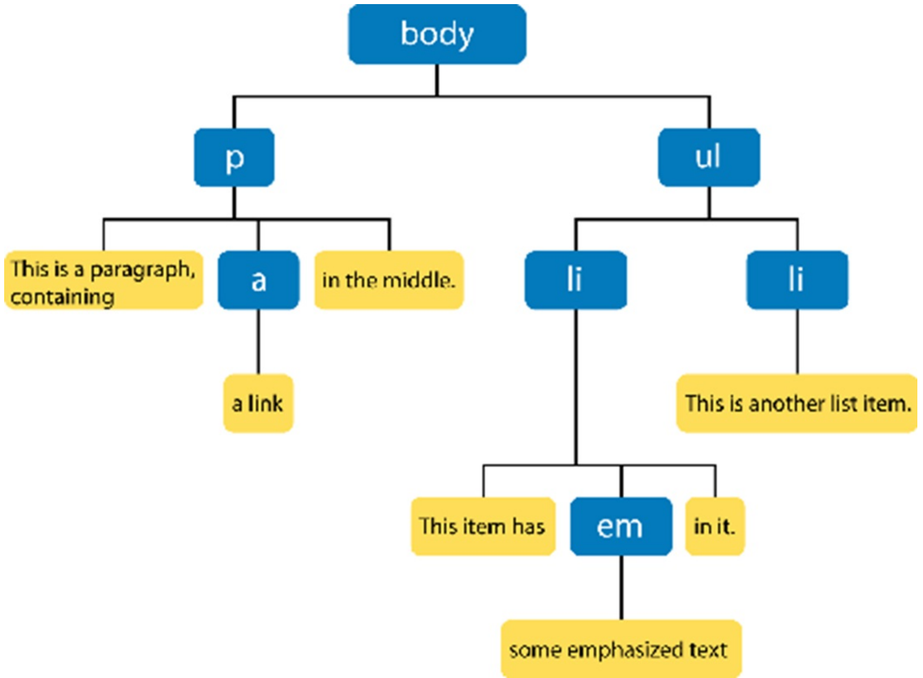
Document Object Model (DOM)

- DOM jest interfejsem pozwalającym na pracę z dokumentami HTML, XML i SVG.
- Dzięki niemu możemy w bardziej zaawansowany sposób komunikować się z użytkownikiem (poprzez wczytywanie zawartości inputów albo dodawanie nowych elementów do strony).

- Przedstawia dokumenty w przystępnej formie drzewa i pozwala na ich manipulację.
- **Dokument musi być w całości załadowany przed przystąpieniem do wykonywania na nim operacji.**

Obiekt document

- Obiekt document jest specjalnym elementem reprezentującym naszą stronę internetową (cały DOM).
- Od niego powinniśmy zacząć wyszukiwanie jakiegokolwiek elementu znajdującego się na stronie.
- Jest on dostępny na stronie od samego początku działania naszego skryptu – nie musimy go ani sami tworzyć, ani wczytywać.



Element

- Podstawowym narzędziem pracy z DOM jest element reprezentujący tag HTML.
- Dzięki temu narzędziu możemy wpływać na wybrane przez nas tagi, np. zmieniać ich zawartość, klasy HTML itp.

- Element drzewa DOM w JavaScript reprezentowany jest przez specjalny obiekt elementu, posiada on atrybuty i metody jak każdy inny obiekt.

Kod HTML

```
<a href="http://www.google.com" class="foo bar" id="glink" data-foo="1">
    <h1>Google</h1>
</a>
```

Kod JavaScript

```
var link = document.querySelector('#glink');
```

Wyszukiwanie elementów w DOM-ie

Wyszukiwanie elementów w DOM-ie

Do wyszukiwania pojedynczego elementu na stronie mamy następujące metody:

```
document.querySelector("selector");
```

```
document.getElementById("id");
```

Metody te zwracają pojedynczy **element** lub **null**, jeśli żaden z elementów nie spełnia wymagań.

Wyszukiwanie elementów w DOM-ie

Do wyszukiwania pojedynczego elementu na stronie mamy następujące metody:

```
document.querySelector("selector");
```

querySelector – wyszukuje **pierwszy** element odpowiadający zapytaniu CSS

```
document.getElementById("id");
```

Metody te zwracają pojedynczy **element** lub **null**, jeśli żaden z elementów nie spełnia wymagań.

Wyszukiwanie elementów w DOM-ie

Do wyszukiwania pojedynczego elementu na stronie mamy następujące metody:

```
document.querySelector("selector");
```

```
document.getElementById("id");
```

getElementById – wyszukuje element z danym **ID**, nie używamy tutaj selektora css a samej nazwy ID.

Metody te zwracają pojedynczy **element** lub **null**, jeśli żaden z elementów nie spełnia wymagań.

Wyszukiwanie elementów w DOM-ie

Do wyszukiwania wielu elementów na stronie mamy następujące metody:

```
document.querySelectorAll("selector");
```

```
document.getElementsByTagName("tag");
```

```
document.getElementsByClassName("className");
```

Wyszukiwanie elementów w DOM-ie

Do wyszukiwania wielu elementów na stronie mamy następujące metody:

```
document.querySelectorAll("selector");
```

querySelectorAll– wyszukuje **wszystkie** elementy odpowiadające zapytaniu CSS

```
document.getElementsByTagName("tag");
```

```
document.getElementsByClassName("className");
```

Wyszukiwanie elementów w DOM-ie

Do wyszukiwania wielu elementów na stronie mamy następujące metody:

```
document.querySelectorAll("selector");
```

```
document.getElementsByTagName("tag");
```

getElementsByTagName - wyszukuje **wszystkie** elementy po podanym tagu, nie używamy tutaj selektora css a samej nazwy tagu

```
document.getElementsByClassName("className");
```

Wyszukiwanie elementów w DOM-ie

Do wyszukiwania wielu elementów na stronie mamy następujące metody:

```
document.querySelectorAll("selector");
```

```
document.getElementsByTagName("tag");
```

```
document.getElementsByClassName("className");
```

getElementsByClassName– wyszukuje **wszystkie** elementy po nazwie klasy, nie używamy tutaj selektora css a samej nazwy klasy

Wyszukiwanie elementów w DOM-ie

Metody te zawsze zwracają tablicę elementów. Jeśli żaden element nie spełnia wymagań – tablica jest pusta.

```
var foo = document.getElementsByClassName("nieIstKlasa");
foo.length;
```

Wyszukiwanie elementów w DOM-ie

Metody te zawsze zwracają tablicę elementów. Jeśli żaden element nie spełnia wymagań – tablica jest pusta.

```
var foo = document.getElementsByClassName("nieIstKlasa");
foo.length;
```

0 elementów w tablicy - brak elementów o klasie **nieIstKlasa** na stronie.

Wyszukiwanie elementów w DOM-ie

Jak łatwo zapamiętać kiedy użyć selektora CSS a kiedy nie?

```
document.querySelectorAll("#home .cup");
```

Jak łatwo zapamiętać kiedy użyć selektora css a kiedy nie?

```
document.getElementsByTagName("div");
```

Wyszukiwanie elementów w DOM-ie

Jak łatwo zapamiętać kiedy użyć selektora CSS a kiedy nie?

```
document.querySelectorAll("#home .cup");
```

querySelectorAll - Jeśli metoda zaczyna się od **query** jako argument przyjmuje ona zawsze **selektor CSS**.

Jak łatwo zapamiętać kiedy użyć selektora css a kiedy nie?

```
document.getElementsByTagName("div");
```

Wyszukiwanie elementów w DOM-ie

Jak łatwo zapamiętać kiedy użyć selektora CSS a kiedy nie?

```
document.querySelectorAll("#home .cup");
```

Jak łatwo zapamiętać kiedy użyć selektora css a kiedy nie?

```
document.getElementsByTagName("div");
```

getElementsByTagName - Jeśli metoda zaczyna się od get jako argument przyjmuje ona **string** będący np. **nazwą klasy, id lub tagu html**

Wyszukiwanie elementów w DOM-ie

Metod tych możemy używać zarówno w obiekcie document, jak i w poszczególnych elementach (wtedy szukamy tylko wewnątrz tego elementu).

```
var myButton = document.querySelector("div .btn");
var allParagraphs = document.querySelectorAll("p");
var barTable = document.getElementsByClassName("bar");
var foo = document.getElementById("glink");
var fooHeader = foo.querySelector("h1");
var stars = document.querySelectorAll("div.star");
var firstStarImage = stars.querySelector("img");//BŁĄD
```

Wyszukiwanie elementów w DOM-ie

Metod tych możemy używać zarówno w obiekcie document, jak i w poszczególnych elementach (wtedy szukamy tylko wewnątrz tego elementu).

```
var myButton = document.querySelector("div .btn");
var allParagraphs = document.querySelectorAll("p");
var barTable = document.getElementsByClassName("bar");
var foo = document.getElementById("glink");
var fooHeader = foo.querySelector("h1");
var stars = document.querySelectorAll("div.star");
var firstStarImage = stars.querySelector("img");//BŁĄD
```

Co oznacza ten selektor?

Wyszukiwanie elementów w DOM-ie

Metod tych możemy używać zarówno w obiekcie document, jak i w poszczególnych elementach (wtedy szukamy tylko wewnątrz tego elementu).

```
var myButton = document.querySelector("div .btn");
var allParagraphs = document.querySelectorAll("p");
var barTable = document.getElementsByClassName("bar");
var foo = document.getElementById("glink");
var fooHeader = foo.querySelector("h1");
var stars = document.querySelectorAll("div.star");
var firstStarImage = stars.querySelector("img");//BŁĄD
```

Co oznacza ten selektor?

Wyszukiwanie elementów w DOM-ie

Metod tych możemy używać zarówno w obiekcie document, jak i w poszczególnych elementach (wtedy szukamy tylko wewnątrz tego elementu).

```
var myButton = document.querySelector("div .btn");
var allParagraphs = document.querySelectorAll("p");
var barTable = document.getElementsByClassName("bar");
var foo = document.getElementById("glink");
var fooHeader = foo.querySelector("h1");
var stars = document.querySelectorAll("div.star");
var firstStarImage = stars.querySelector("img");//BŁĄD
```

Co oznacza ten selektor?

Wyszukiwanie elementów w DOM-ie

Metod tych możemy używać zarówno w obiekcie document, jak i w poszczególnych elementach (wtedy szukamy tylko wewnątrz tego elementu).

```
var myButton = document.querySelector("div .btn");
var allParagraphs = document.querySelectorAll("p");
var barTable = document.getElementsByClassName("bar");
var foo = document.getElementById("glink");
var fooHeader = foo.querySelector("h1");
var stars = document.querySelectorAll("div.star");
var firstStarImage = stars.querySelector("img");//BŁĄD
```

Szukamy elementu o id **glink**

Wyszukiwanie elementów w DOM-ie

Metod tych możemy używać zarówno w obiekcie document, jak i w poszczególnych elementach (wtedy szukamy tylko wewnątrz tego elementu).

```
var myButton = document.querySelector("div .btn");
var allParagraphs = document.querySelectorAll("p");
var barTable = document.getElementsByClassName("bar");
var foo = document.getElementById("glink");
var fooHeader = foo.querySelector("h1");
var stars = document.querySelectorAll("div.star");
var firstStarImage = stars.querySelector("img");//BŁĄD
```

Szukamy elementu **h1** we wcześniej znalezionym elemencie

Wyszukiwanie elementów w DOM-ie

Metod tych możemy używać zarówno w obiekcie document, jak i w poszczególnych elementach (wtedy szukamy tylko wewnątrz tego elementu).

```
var myButton = document.querySelector("div .btn");
var allParagraphs = document.querySelectorAll("p");
var barTable = document.getElementsByClassName("bar");
var foo = document.getElementById("glink");
var fooHeader = foo.querySelector("h1");
var stars = document.querySelectorAll("div.star");
var firstStarImage = stars.querySelector("img");//BŁĄD
```

UWAGA: wyszukiwać względnie możemy TYLKO w pojedynczym elemencie, tutaj próbujemy szukać w tablicy elementów - **stars**

Zadania

Podstawowe informacje o elementach

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać. Oto najważniejsze z nich:

- **classList** – zwraca listę klas HTML, jako tablica
- **className** – zwraca lub nastawia nazwy klas **HTML** jako napis
- **id** – zwraca lub nastawia atrybut **ID HTML** jako napis
- **innerHTML** – zwraca lub nastawia kod **HTML** znajdujący się **w** tagu.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.classList;//pobieramy
link.className;//pobieramy
link.id;//pobieramy
link.innerHTML;//pobieramy
link.id = 'newId';
link.className = 'newClass1 newClass2';
```

Atrybuty mają tylko pojedyncze elementy, nie możemy ich używać na tablicy elementów.

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać. Oto najważniejsze z nich:

- **classList** – zwraca listę klas HTML, jako tablica
- **className** – zwraca lub nastawia nazwy klas **HTML** jako napis
- **id** – zwraca lub nastawia atrybut **ID HTML** jako napis
- **innerHTML** – zwraca lub nastawia kod **HTML** znajdujący się **w** tagu.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.classList;//pobieramy
link.className;//pobieramy
link.id;//pobieramy
link.innerHTML;//pobieramy
link.id = 'newId';
link.className = 'newClass1 newClass2';
```

```
["foo", "bar"]
```

Atrybuty mają tylko pojedyncze elementy, nie możemy ich używać na tablicy elementów.

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać. Oto najważniejsze z nich:

- **classList** – zwraca listę klas HTML, jako tablica
- **className** – zwraca lub nastawia nazwy klas **HTML** jako napis
- **id** – zwraca lub nastawia atrybut **ID HTML** jako napis
- **innerHTML** – zwraca lub nastawia kod **HTML** znajdujący się **w** tagu.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.classList;//pobieramy
link.className;//pobieramy
link.id;//pobieramy
link.innerHTML;//pobieramy
link.id = 'newId';
link.className = 'newClass1 newClass2';
```

"foo bar"

Atrybuty mają tylko pojedyncze elementy, nie możemy ich używać na tablicy elementów.

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać. Oto najważniejsze z nich:

- **classList** – zwraca listę klas HTML, jako tablica
- **className** – zwraca lub nastawia nazwy klas **HTML** jako napis
- **id** – zwraca lub nastawia atrybut **ID HTML** jako napis
- **innerHTML** – zwraca lub nastawia kod **HTML** znajdujący się **w** tagu.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.classList;//pobieramy
link.className;//pobieramy
link.id;//pobieramy
link.innerHTML;//pobieramy
link.id = 'newId';
link.className = 'newClass1 newClass2';
```

"glink"

Atrybuty mają tylko pojedyncze elementy, nie możemy ich używać na tablicy elementów.

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać. Oto najważniejsze z nich:

- **classList** – zwraca listę klas HTML, jako tablica
- **className** – zwraca lub nastawia nazwy klas **HTML** jako napis
- **id** – zwraca lub nastawia atrybut **ID HTML** jako napis
- **innerHTML** – zwraca lub nastawia kod **HTML** znajdujący się **w** tagu.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.classList;//pobieramy
link.className;//pobieramy
link.id;//pobieramy
link.innerHTML;//pobieramy
link.id = 'newId';
link.className = 'newClass1 newClass2';
```

"<h1>WP.PL</h1>"

Atrybuty mają tylko pojedyncze elementy, nie możemy ich używać na tablicy elementów.

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać. Oto najważniejsze z nich:

- **classList** – zwraca listę klas HTML, jako tablica
- **className** – zwraca lub nastawia nazwy klas **HTML** jako napis
- **id** – zwraca lub nastawia atrybut **ID HTML** jako napis
- **innerHTML** – zwraca lub nastawia kod **HTML** znajdujący się **w** tagu.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.classList;//pobieramy
link.className;//pobieramy
link.id;//pobieramy
link.innerHTML;//pobieramy
link.id = 'newId';
link.className = 'newClass1 newClass2';
```

nadpisujemy id nową wartością

Atrybuty mają tylko pojedyncze elementy, nie możemy ich używać na tablicy elementów.

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać. Oto najważniejsze z nich:

- **classList** – zwraca listę klas HTML, jako tablica
- **className** – zwraca lub nastawia nazwy klas **HTML** jako napis
- **id** – zwraca lub nastawia atrybut **ID HTML** jako napis
- **innerHTML** – zwraca lub nastawia kod **HTML** znajdujący się **w** tagu.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.classList;//pobieramy
link.className;//pobieramy
link.id;//pobieramy
link.innerHTML;//pobieramy
link.id = 'newId';
link.className = 'newClass1 newClass2';
```

nadpisujemy listę klas nową wartością

Atrybuty mają tylko pojedyncze elementy, nie możemy ich używać na tablicy

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać, oto najważniejsze z nich:

- **outerHTML** – zwraca/nastawia kod HTML wraz z tagiem,
- **innerText** – – zwraca/nastawia tekst znajdujący się w tagu (bez zagnieżdżonych tagów),
- **tagName** – zwraca nazwę tagu
- **dataset** – zwraca tablicę asocjacyjną dataset.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać, oto najważniejsze z nich:

- **outerHTML** – zwraca/nastawia kod HTML wraz z tagiem,
- **innerText** – – zwraca/nastawia tekst znajdujący się w tagu (bez zagnieżdżonych tagów),
- **tagName** – zwraca nazwę tagu
- **dataset** – zwraca tablicę asocjacyjną dataset.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
"<a href="http://wp.pl" class="foo
bar" id="glink" data-foo="1">
<h1>WP.PL</h1> </a>"
```

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać, oto najważniejsze z nich:

- **outerHTML** – zwraca/nastawia kod HTML wraz z tagiem,
- **innerText** – – zwraca/nastawia tekst znajdujący się w tagu (bez zagnieżdżonych tagów),
- **tagName** – zwraca nazwę tagu
- **dataset** – zwraca tablicę asocjacyjną dataset.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
"WP.PL"
```

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać, oto najważniejsze z nich:

- **outerHTML** – zwraca/nastawia kod HTML wraz z tagiem,
- **innerText** – – zwraca/nastawia tekst znajdujący się w tagu (bez zagnieżdżonych tagów),
- **tagName** – zwraca nazwę tagu
- **dataset** – zwraca tablicę asocjacyjną dataset.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

"A" – nazwa zwracana jest dużymi literami

Atrybuty elementu

Element ma kilka podstawowych atrybutów, które możemy zmieniać, oto najważniejsze z nich:

- **outerHTML** – zwraca/nastawia kod HTML wraz z tagiem,
- **innerText** – – zwraca/nastawia tekst znajdujący się w tagu (bez zagnieżdżonych tagów),
- **tagName** – zwraca nazwę tagu
- **dataset** – zwraca tablicę asocjacyjną dataset.

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

```
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
{ foo: "1" }
```

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
["foo", "bar"]
```

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
"foo bar"
```

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
"glink"
```

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

"<h1>WP.PL</h1>"

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

"<a href...> <h1>WP.PL</h1> "

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
"WP.PL"
```

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
"A"
```

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

```
{ foo: "1" }
```

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

Atrybuty elementu – podsumowanie

Kod HTML

```
<a href="http://wp.pl" class="foo bar"
   id="glink" data-foo="1">
    <h1>WP.PL</h1>
</a>
```

Kod Javascript

```
var link =
document.querySelector("#glink");
```

Atrybuty, z którymi będziesz za pan brat

```
link.classList;
link.className;
link.id;
link.innerHTML;
link.outerHTML;
link.innerText;
link.tagName;
link.dataset;
```

Pamiętaj, że wszystkie te atrybuty możesz również ustawiać

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

Zadania

Więcej o elementach

style

- Każdy pojedynczy element posiada atrybut **`style`**, będący obiektem zawierającym style.
- Obiekt **`style`** przechowuje wszystkie wartości jako **stringi (napisy)**.
- Tak samo będą one nam zwracane i tak powinniśmy je nastawiać.

Aktualną wartość stylu możemy wczytać:

```
element.style.backgroundColor;
element.style.fontSize;
element.style.padding;
```

Albo nastawić nową wartość:

```
element.style.backgroundColor = "blue";
```

style

- Każdy pojedynczy element posiada atrybut **`style`**, będący obiektem zawierającym style.
- Obiekt **`style`** przechowuje wszystkie wartości jako **stringi (napisy)**.
- Tak samo będą one nam zwracane i tak powinniśmy je nastawiać.

Aktualną wartość stylu możemy wczytać:

```
element.style.backgroundColor;
element.style.fontSize;
element.style.padding;
```

Nazwy własności obiektu **`style`** są tożsame z własnościami css ale zamienionymi na **`camelCase`**

Albo nastawić nową wartość:

```
element.style.backgroundColor = "blue";
```

style

- Każdy pojedynczy element posiada atrybut **`style`**, będący obiektem zawierającym style.
- Obiekt **`style`** przechowuje wszystkie wartości jako **stringi (napisy)**.
- Tak samo będą one nam zwracane i tak powinniśmy je nastawiać.

Aktualną wartość stylu możemy wczytać:

```
element.style.backgroundColor;
element.style.fontSize;
element.style.padding;
```

Obiekt **`style`** zna jedynie style ustawione przez ten obiekt, nie zna on styli css elementu

Albo nastawić nową wartość:

```
element.style.backgroundColor = "blue";
```

classList

Metoda **classList** elementu zwraca listę wszystkich klas tego elementu. Możemy łatwo z nią pracować dzięki następującym metodom:

- **el.classList.add(className)** – dodaje podaną klasę,
- **el.classList.remove(className)** – usuwa podaną klasę,
- **el.classList.toggle(className)** – przełącza podaną klasę (czyli usuwa jeżeli jest, jeżeli jej nie ma, to dodaje).

classList

Mamy taki element:

Kod HTML

```
<div id="myDiv" class="class1 class2">
</div>
```

Kod Javascript

```
var myDiv =
document.getElementById("myDiv");
```

Możemy łatwo wczytać jego wszystkie klasy:

```
console.log(myDiv.classList);
console.log(myDiv.className);
```

classList

Mamy taki element:

Kod HTML

```
<div id="myDiv" class="class1 class2">
</div>
```

Kod Javascript

```
var myDiv =
document.getElementById("myDiv");
```

Możemy łatwo wczytać jego wszystkie klasy:

```
console.log(myDiv.classList);
console.log(myDiv.className);
```

["class1", "class2"] -> tablica

classList

Mamy taki element:

Kod HTML

```
<div id="myDiv" class="class1 class2">
</div>
```

Kod Javascript

```
var myDiv =
document.getElementById("myDiv");
```

Możemy łatwo wczytać jego wszystkie klasy:

```
console.log(myDiv.classList);
console.log(myDiv.className);
```

"class1 class2" -> string

classList

Możemy dodać nową klasę:

```
myDiv.classList.add("nowaKlasa");
console.log(myDiv.classList);
```

Możemy usunąć jedną z jego klas:

```
myDiv.classList.remove("class1");
console.log(myDiv.classList);
```

Możemy przełączać daną klasę:

```
myDiv.classList.toggle("toggleClass1");
```

```
myDiv.classList.toggle("nowaKlasa");
```

```
console.log(myDiv.classList);
```

classList jest lepsze niż **className** do dodawania klas, ponieważ **className** nadpisuje aktualną listę klas nową wartością a **classList** do niej dodaje

classList

Możemy dodać nową klasę:

```
myDiv.classList.add("nowaKlasa");
console.log(myDiv.classList);
```

```
["class1", "class2", "nowaKlasa"]
```

Możemy usunąć jedną z jego klas:

```
myDiv.classList.remove("class1");
console.log(myDiv.classList);
```

Możemy przełączać daną klasę:

```
myDiv.classList.toggle("toggleClass1");
```

```
myDiv.classList.toggle("nowaKlasa");
```

```
console.log(myDiv.classList);
```

classList jest lepsze niż **className** do dodawania klas, ponieważ **className** nadpisuje aktualną listę klas nową wartością a **classList** do niej dodaje

classList

Możemy dodać nową klasę:

```
myDiv.classList.add("nowaKlasa");
console.log(myDiv.classList);
```

Możemy usunąć jedną z jego klas:

```
myDiv.classList.remove("class1");
console.log(myDiv.classList);
```

["class2", "nowaKlasa"]

Możemy przełączać daną klasę:

```
myDiv.classList.toggle("toggleClass1");
```

```
myDiv.classList.toggle("nowaKlasa");
```

```
console.log(myDiv.classList);
```

classList jest lepsze niż **className** do dodawania klas, ponieważ **className** nadpisuje aktualną listę klas nową wartością a **classList** do niej dodaje

classList

Możemy dodać nową klasę:

```
myDiv.classList.add("nowaKlasa");
console.log(myDiv.classList);
```

Możemy usunąć jedną z jego klas:

```
myDiv.classList.remove("class1");
console.log(myDiv.classList);
```

Możemy przełączać daną klasę:

```
myDiv.classList.toggle("toggleClass1");
```

dodaje ponieważ klasa nie istnieje

```
myDiv.classList.toggle("nowaKlasa");
```

```
console.log(myDiv.classList);
```

`classList` jest lepsze niż **`className`** do dodawania klas, ponieważ **`className`** nadpisuje aktualną listę klas nową wartością a **`classList`** do niej dodaje

classList

Możemy dodać nową klasę:

```
myDiv.classList.add("nowaKlasa");
console.log(myDiv.classList);
```

Możemy usunąć jedną z jego klas:

```
myDiv.classList.remove("class1");
console.log(myDiv.classList);
```

Możemy przełączać daną klasę:

```
myDiv.classList.toggle("toggleClass1");
```

```
myDiv.classList.toggle("nowaKlasa");
```

usuwa ponieważ klasa istnieje

```
console.log(myDiv.classList);
```

classList jest lepsze niż **className** do dodawania klas, ponieważ **className** nadpisuje aktualną listę klas nową wartością a **classList** do niej dodaje

classList

Możemy dodać nową klasę:

```
myDiv.classList.add("nowaKlasa");
console.log(myDiv.classList);
```

Możemy usunąć jedną z jego klas:

```
myDiv.classList.remove("class1");
console.log(myDiv.classList);
```

Możemy przełączać daną klasę:

```
myDiv.classList.toggle("toggleClass1");
```

```
myDiv.classList.toggle("nowaKlasa");
```

```
console.log(myDiv.classList);
```

```
["class2", "toggleClass1"]
```

classList jest lepsze niż **className** do dodawania klas, ponieważ **className** nadpisuje aktualną listę klas nową wartością a **classList** do niej dodaje

dataset

Dane powiązane z tagiem

Możemy przetrzymać pewne dane powiązane z tagiem HTML, które mogą nam się później przydać, na przykład:

- tagi do zdjęcia,
- tooltip,
- ID obiektu.

Takie dane powinniśmy trzymać w specjalnym atrybucie zaczynającym się od **`data-`**

- W JavaScript mamy dostęp do specjalnego obiektu **`dataset`** należącego do elementu.
- Dzięki niemu możemy nastawiać lub wczytywać informacje z datasetu.

dataset

Wszystkie dane poniższego elementu możemy z łatwością wczytać z datasetu:

Kod HTML

```
<div id="user" data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset);
console.log(myUser.dataset.id);
console.log(myUser.dataset.user);
console.log(myUser.dataset.dateOfBirth);
```

dataset

Wszystkie dane poniższego elementu możemy z łatwością wczytać z datasetu:

Kod HTML

```
<div id="user" data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset);
console.log(myUser.dataset.id);
console.log(myUser.dataset.user);
console.log(myUser.dataset.dateOfBirth);
```

```
{id: "1234567890", user: "johndoe",
dateOfBirth: ""}
```

dataset

Wszystkie dane poniższego elementu możemy z łatwością wczytać z datasetu:

Kod HTML

```
<div id="user" data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset);
console.log(myUser.dataset.id);
console.log(myUser.dataset.user);
console.log(myUser.dataset.dateOfBirth);
```

```
1234567890
```

dataset

Wszystkie dane poniższego elementu możemy z łatwością wczytać z datasetu:

Kod HTML

```
<div id="user" data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset);
console.log(myUser.dataset.id);
console.log(myUser.dataset.user);
console.log(myUser.dataset.dateOfBirth);
```

```
johndoe
```

dataset

Wszystkie dane poniższego elementu możemy z łatwością wczytać z datasetu:

Kod HTML

```
<div id="user" data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset);
console.log(myUser.dataset.id);
console.log(myUser.dataset.user);
console.log(myUser.dataset.dateOfBirth);
```

"" - pusty string

dataset

Zmiana wartości w datasecie

Do istniejącego datasetu możemy przypisywać nową wartość.

Kod HTML

```
<div id="user" data-id="123456"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset.id);
myUser.dataset.id = 4444;
console.log(myUser.dataset.id);
```

dataset

Zmiana wartości w datasecie

Do istniejącego datasetu możemy przypisywać nową wartość.

Kod HTML

```
<div id="user" data-id="123456"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset.id);
myUser.dataset.id = 4444;
console.log(myUser.dataset.id);
```

123456

dataset

Zmiana wartości w datasecie

Do istniejącego datasetu możemy przypisywać nową wartość.

Kod HTML

```
<div id="user" data-id="123456"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset.id);
myUser.dataset.id = 4444;
console.log(myUser.dataset.id);
```

4444

dataset

Nowa wartość w datasecie

Możemy dodawać nowy, nieistniejący wcześniej dataset.

Kod HTML

```
<div id="user"
     data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset.something);
myUser.dataset.something = "new value";
console.log(myUser.dataset.something);
```

dataset

Nowa wartość w datasecie

Możemy dodawać nowy, nieistniejący wcześniej dataset.

Kod HTML

```
<div id="user"
     data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset.something);
myUser.dataset.something = "new value";
console.log(myUser.dataset.something);
```

```
""
```

dataset

Nowa wartość w datasecie

Możemy dodawać nowy, nieistniejący wcześniej dataset.

Kod HTML

```
<div id="user"
     data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe
</div>
```

Kod Javascript

```
var myUser =
document.querySelector("#user");
```

```
console.log(myUser.dataset.something);
myUser.dataset.something = "new value";
console.log(myUser.dataset.something);
```

```
"new value"
```

Atrybuty elementów

Z poziomu JavaScript możemy edytować wszystkie atrybuty danego elementu. Służą do tego metody przedstawione na kolejnych slajdach.

```
<a href="www.google.com" id="glink">Hello Google!</a>
```

```
var link = document.querySelector("#glink");
```

Atrybuty elementów

- **el.hasAttribute(attrName)** – sprawdza, czy element ma podany atrybut. W odpowiedzi dostajemy wartość boolean.

```
link.hasAttribute("href");
```

- **el.getAttribute(attrName)** – zwraca wartość podanego atrybutu.

```
link.getAttribute('href');
```

Atrybuty elementów

- **el.hasAttribute(attrName)** – sprawdza, czy element ma podany atrybut. W odpowiedzi dostajemy wartość boolean.

```
link.hasAttribute("href");
```

true

- **el.getAttribute(attrName)** – zwraca wartość podanego atrybutu.

```
link.getAttribute('href');
```

Atrybuty elementów

- **el.hasAttribute(attrName)** – sprawdza, czy element ma podany atrybut. W odpowiedzi dostajemy wartość boolean.

```
link.hasAttribute("href");
```

- **el.getAttribute(attrName)** – zwraca wartość podanego atrybutu.

```
link.getAttribute('href');
```

"www.google.com"

Atrybuty elementów

- **el.removeAttribute(attrName)** – usuwa podany atrybut.

```
link.removeAttribute("href");
link.hasAttribute("href");
link.getAttribute("href");
```

- **el.setAttribute(attrName, attrValue)** – nastawia wartość podanego atrybutu.

```
link.setAttribute("href", "www.wp.pl");
link.hasAttribute("href");
link.getAttribute("href");
```

Atrybuty elementów

- **el.removeAttribute(attrName)** – usuwa podany atrybut.

```
link.removeAttribute("href");
link.hasAttribute("href");
link.getAttribute("href");
```

false

- **el.setAttribute(attrName, attrValue)** – nastawia wartość podanego atrybutu.

```
link.setAttribute("href", "www.wp.pl");
link.hasAttribute("href");
link.getAttribute("href");
```

Atrybuty elementów

- **el.removeAttribute(attrName)** – usuwa podany atrybut.

```
link.removeAttribute("href");
link.hasAttribute("href");
link.getAttribute("href");
```

null

- **el.setAttribute(attrName, attrValue)** – nastawia wartość podanego atrybutu.

```
link.setAttribute("href", "www.wp.pl");
link.hasAttribute("href");
link.getAttribute("href");
```

Atrybuty elementów

- **el.removeAttribute(attrName)** – usuwa podany atrybut.

```
link.removeAttribute("href");
link.hasAttribute("href");
link.getAttribute("href");
```

- **el.setAttribute(attrName, attrValue)** – nastawia wartość podanego atrybutu.

```
link.setAttribute("href", "www.wp.pl");
link.hasAttribute("href");
link.getAttribute("href");
```

true

Atrybuty elementów

- **el.removeAttribute(attrName)** – usuwa podany atrybut.

```
link.removeAttribute("href");
link.hasAttribute("href");
link.getAttribute("href");
```

- **el.setAttribute(attrName, attrValue)** – nastawia wartość podanego atrybutu.

```
link.setAttribute("href", "www.wp.pl");
link.hasAttribute("href");
link.getAttribute("href");
```

"www.wp.pl"

Zadania