

JavaScript

– DOM

- dzień 1

v 1.5

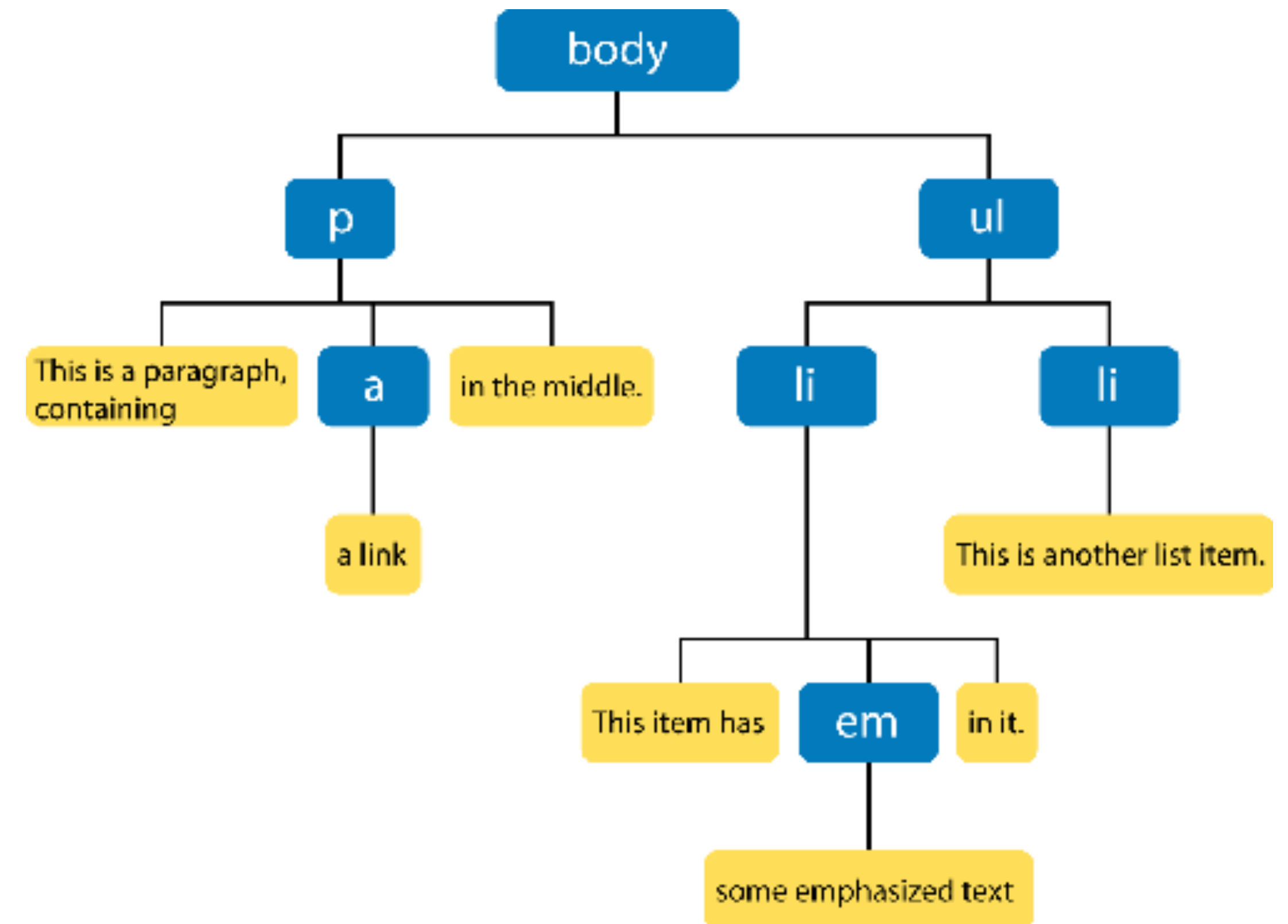
PLAN

- [Wprowadzenie](#)
- [Wyszukiwanie elementów w DOM-ie](#)
- [Elementy](#)
- [Więcej o elemencie](#)

Wprowadzenie

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.



Wyszukiwanie elementów w DOM-ie

Wyszukiwanie elementów w DOM-ie

Do wyszukiwania pojedynczego elementu na stronie mamy następujące metody:

- `document.querySelector("selector")` – wyszukuje pierwszy element odpowiadający zapytaniu CSS,
- `document.getElementById("id")` – wyszukuje element z danym 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.querySelector("selector")`
– wyszukuje wszystkie elementy odpowiadające zapytaniu CSS,
- `document.getElementsByTagName("tag")`
– wyszukuje po podanym tagu,
- `el.getElementsByClassName("className")`
– wyszukuje po nazwie klasy.

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

Wyszukiwanie elementów w DOM-ie

Jak łatwo zapamiętać kiedy użyć selektora CSS a kiedy nie?

➤ `document.querySelector("selector")`



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("tag")`



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).

Szukając wewnątrz elementu, możemy użyć każdej metody z wyjątkiem **getElementById**, która działa jedynie na obiekcie **document**.

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

Zadania

Wykonaj zadania z działu
2.DOM
z folderu

1.Wyszukiwanie_Elementow



Elementy

Atrybuty elementu

Po co szukamy elementów na stronie?

Kiedy już znajdziemy element na stronie możemy z nim zrobić wiele rzeczy np. pobrać listę jego klas, zmodyfikować nazwę klas, zmienić tekst, który wyświetla, zmienić style, pobrać zbiór dataset i wiele innych rzeczy.

Obok kilka przykładów atrybutów, które możemy pobierać lub modyfikować.

Przykładowe atrybuty

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

Atrybuty elementu - przykład

Kod HTML

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

Kod JavaScript

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

```
link.className; // "foo bar"
```

```
link.id; // "glink"
```

```
link.innerHTML; // "<h1>Google</h1>"
```

```
link.outerHTML; // "<a href='www.google.com' class='foo bar'
                  id='glink' data-foo='1'><h1>Google</h1></a>"
```

```
link.innerText; // "Google"
```

```
link.tagName; // "A" – nazwa zwracana jest dużymi literami
```

```
link.dataset; // { foo: "1" }
```

Atrybuty elementu – podsumowanie

Kod HTML

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

Kod JavaScript

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

Atrybuty, z którymi będziesz za pan brat

```
link.classList; // ["foo", "bar"]
link.className; // "foo bar"
link.id; // "glink"
link.innerHTML; // "<h1>Google</h1>"
link.outerHTML;
// "<a href... ><h1>Google</h1></a>"
link.innerText; // "Google"
link.tagName; // "A"
link.dataset; // { foo: "1" }
```

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

dataset

W zadaniach będziemy korzystać z atrybutu dataset. Dlatego zrobmy krótkie wprowadzenie czym jest ten atrybut.

Porozmawiamy o nim dokładniej w kolejnym rozdziale

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-**

Więcej:

https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_data_attributes

- 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**:

Wyświetlanie

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

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

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

Zadania

Wykonaj zadania
z działu
2.DOM
z folderu
2.Element



Więcej o elemencie



classList

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 - przykład

Mamy taki element:

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

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

Możemy łatwo wczytać jego wszystkie klasy:

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

```
// ["class1", "class2"] <- obiekt
```

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

```
// class1 class2 <- string
```

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);  
// ["class2", "nowaKlasa"]
```

Możemy przełączać daną klasę:

```
//dodaje ponieważ klasa nie istnieje  
myDiv.classList.toggle("toggleClass1");  
//usuwa ponieważ klasa istnieje  
myDiv.classList.toggle("nowaKlasa");  
  
console.log(myDiv.classList);  
//["class2", "toggleClass1"]
```



dataset


dataset

Dane powiązane z tagiem

Wiesz już czym jest dataset i jak wyświetlić jego wartości. Sprawdźmy teraz jak możemy jest ustawiać i zmieniać.

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

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

```
console.log(myUser.dataset);  
// {id: "1234567890", user: "johndoe", dateOfBirth: ""}  
  
console.log(myUser.dataset.id);  
// 1234567890  
  
console.log(myUser.dataset.user);  
// johndoe  
  
console.log(myUser.dataset.dateOfBirth);  
  
//  Pusty element
```

dataset

Zmiana wartości w datasetcie

Do istniejącego **datasetu** możemy przypisywać nową wartość.

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

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

```
console.log(myUser.dataset.id); // 123456
```

```
myUser.dataset.id = 4444;
```

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

← Stara wartość

← Nowa wartość

dataset

Nowy wartość w datasecie

Możemy dodawać nowy,
nieistniejący wcześniej dataset.

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

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

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

Atrybuty - metody

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"); // true
```

- **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"); // false
```

```
link.getAttribute("href"); // null
```

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

```
link.setAttribute("href", "www.something.com");
```

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

```
link.getAttribute("href"); // "www.something.com"
```

Obiekt style

Pobieranie i modyfikacja stylów CSS

- Obiekt `style` przechowuje wszystkie wartości jako stringi (napisy).
- Tak samo będą one nam zwracane i tak powinniśmy je nastawiać.
- Obiekt `style` "widzi" tylko style ustawione za pomocą JavaScript, nie widzi stylów CSS.

Aktualną wartość stylu możemy wczytać:

```
element.style.backgroundColor;
```

Albo nastawić nową wartość:

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

Zadania

Wykonaj zadania
z działu
2.DOM
z folderu
3.Wiecej_o_elemencie