JavaScript podstawy – DOM

v 1.3

Coders Lab
SZKOŁA PROGRAMOWANIA

PLAN

- Wprowadzenie
- Wyszukiwanie elementów w DOM-ie
- Więcej o elementach
- Eventy w DOM-ie
- Poruszanie się po drzewie DOM
- Inputy i formularze

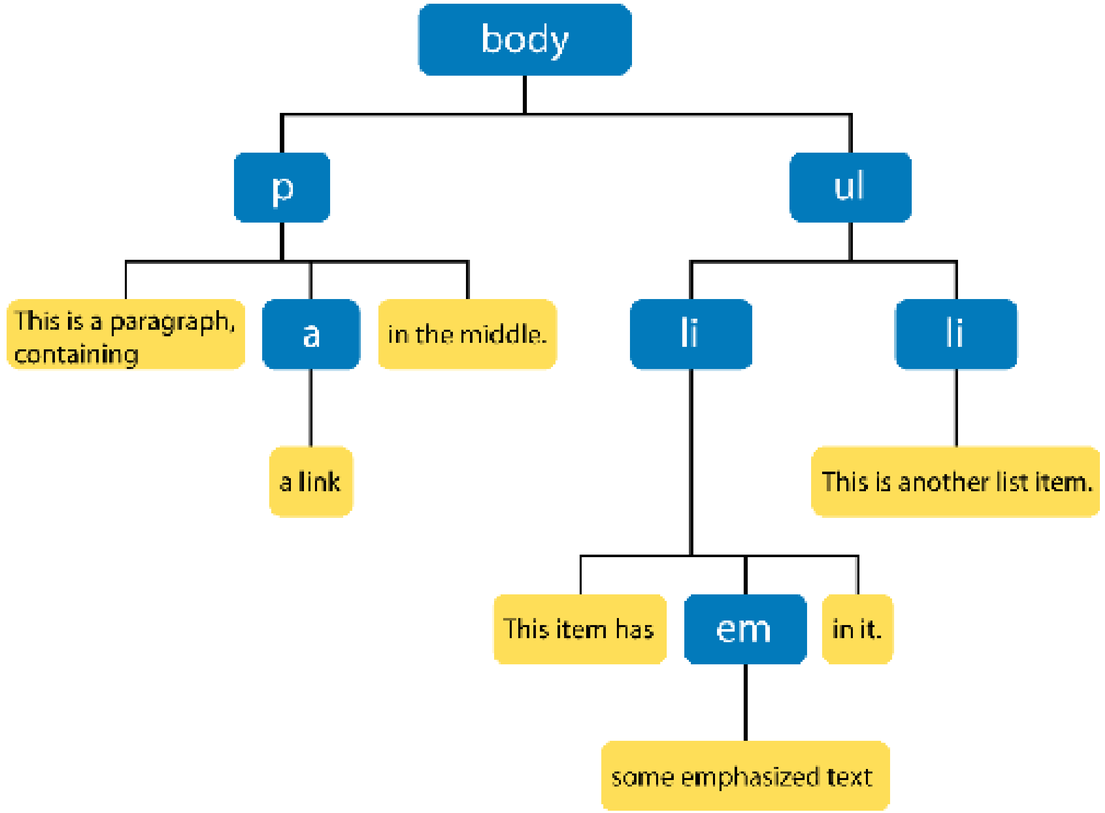
Wprowadzenie

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.

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`);
```

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.querySelectorAll("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**.

```
var foo =

document.getElementsByClassName("nieIstKlasa");

foo.length; // 0
```

Wyszukiwanie elementów w DOM-ie

Jak łatwo zapamiętać kiedy użyć selektora CSS a kiedy nie?

- **document.querySelectorAll("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).

```
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");
```

Czas na zadania

Wykonajcie zadania z działu:

2. DOM

Katalog

2. Wyszukiwanie elementów

Podstawowe informacje o elementach

Atrybuty elementu

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

- **classList** – zwraca listę 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.

```
link.classList; // ["foo", "bar"]

link.className; // "foo bar"

link.id; // "glink"

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

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**.

```
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

Czas na zadania

Wykonajcie zadania z działu:

2. DOM

Katalog

1. Element

Więcej o elementach

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;
```

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:

```
<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

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**:

```
<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 datasecie

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 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"
```

Czas na zadania

Wykonajcie zadania z działu:

2. DOM

Katalog

3. Więcej o elemencie