Rodzaje uchwytów elementów DOM

Uchwyt (*ang. reference*) to inaczej zmienna (wskaźnik, referencja), która przechowuje odwołanie do konkretnego elementu HTML w dokumencie. Mówiąc prościej – jeśli chcemy, aby nasz skrypt JS mógł zmienić coś w elemencie HTML (np. odczytać tekst z pola formularza, sprawdzić czy checkbox jest zaznaczony, albo zmienić kolor przycisku czy inny styl), to najpierw musimy ten element w ogóle jakoś "złapać" do edycji w kodzie. JavaScript daje nam do wyboru wiele metod uzyskiwania dostępu do obiektów DOM (*ang. Document Object Model*), a każda z nich sprawdza się w różnych sytuacjach.

Dzięki uchwytom w JavaScript:

- wskazujemy konkretne elementy, które nas interesują,
- zyskujemy nad nimi pełną kontrolę (możemy je modyfikować, ukrywać, zmieniać style, treść itd.),
- skrypt działa tylko na tych wybranych elementach strony, bez "ruszania" innych.

Na egzaminie INF.03 bardzo często pracujemy z formularzami, walidacją wprowadzonych danych, modyfikacją stylów lub dynamiczną zmianą treści elementów. I za każdym razem kluczową rolę odgrywa uchwyt, bo tylko dzięki niemu jesteśmy w stanie:

- pobrać dane z pola tekstowego,
- sprawdzić zaznaczenie pola checkbox,
- pokazać wynik obliczeń w akapicie czy divie,
- zmienić kolor tekstu.

Poznajmy teraz wiele rodzajów uchwytów, aby dysponować sporym wachlarzem możliwości na egzaminie.

getElementById() – pobranie uchwytu elementu identyfikatorem

Najprostsza i często używana metoda to pobranie elementu po jego

`id` – ten identyfikator powinien być unikalny w całym dokumencie HTML. Załóżmy, że w polu tekstowym formularza podajemy imię:

```
<input type="text" id="imie" placeholder="Wpisz swoje imię">
<button onclick="pobierzImie()">Pobierz wartość</button>
<p id="wynik"></p>
```

Odczytu tej wprowadzonej wartości łatwo dokonać po uchwyceniu konkretnego pola po identyfikatorze:

```
function pobierzImie() {
    let input = document.getElementById("imie");
    let wynik = document.getElementById("wynik");
    wynik.innerHTML = "Wpisane imię: " + input.value;
}
```

Natomiast oczywiście istnienie identyfikatora (pilnowanie unikalności) zawsze zmusza przeglądarkę do nieco większego wysiłku pamięciowego aniżeli istnienie klas czy po prostu tagów. Dlatego też mimo popularności uchwytu nie zawsze chcemy dużo identyfikatorów w dokumencie definiować – często wystarczy nam klasa, sam znacznik albo

querySelector() – uchwycenie elementu selektorem CSS

Uchwyt ten służy do pobierania jednego, pierwszego pasującego elementu w strukturze HTML. W przeciwieństwie do

`getElementById()`, które działa tylko z ustanowionym atrybutem `id` w tagu, tutaj możemy zastosować silnik selektorów CSS – a więc oprócz identyfikatora złapać także tag, klasę, pseudoklasę, a nawet selektor złożony (np. `#containter p`) – zupełnie tak jakbyśmy definiowali selektor elementu w CSS. Dlatego też jest to świetny uchwyt do stosowania na egzaminie.

Załóżmy, że istnieją na stronie dwa akapity o następującej klasie:

```
<p class="tekst">Lorem ipsum dolor sit amet.</p>
<p class="tekst">Nullam vulputate dolor non purus tristique, vel rutrum nulla dictum.
</p>
<button onclick="zmienKolor()">Zmień kolor na niebieski</button>
```

I teraz rozważmy następujący kod skryptu JS:

```
function zmienKolor() {
    document.querySelector(".tekst").style.color = "blue";
}
```

Co ważne – zapis

`querySelector(".tekst")`, złapie pierwszy znaleziony paragraf o tej klasie na stronie – i tylko ten jeden. Czyli jedynie pierwszy akapit stanie się niebieski po kliknięciu na przycisk. Jeśli chcemy dobrać się do wielu elementów jednocześnie, to musimy użyć innego selektora – `querySelectorAll()`. Ta metoda zwróci tzw. *NodeList* (*ang. lista węzłów*), czyli coś podobnego do tablicy, po której możemy iterować pętlą.

querySelectorAll() – uchwycenie kolekcji elementów selektorem CSS

Aby pobrać wszystkie pasujące do selektora elementy, używamy

`querySelectorAll()`, który zwraca `NodeList`. Załóżmy np. istnienie kilku linków w stopce witryny:

```
<footer>
    <a href="#">Polityka prywatności</a>
    <a href="#">Regulamin</a>
    <a href="#">Kontakt</a>
</footer>
```

Pracując na kolekcji elementów zamiast na pojedynczym obiekcie (złapaliśmy kolekcję tym razem po znacznikach), nadajemy im kolejno w skrypcie od razu szary kolor:

```
let linkiStopki = document.querySelectorAll("footer a");
linkiStopki.forEach(function(link) {
    link.style.color = "gray";
});
```

Użyliśmy tutaj metody

`forEach`, dedykowanej do zapętlonej pracy z elementami kolekcji oraz funkcji anonimowej – pojedynczy uchwycony obiekt nazywamy w iteracji `link` i zmieniamy jego kolor na szary. Oczywiście równie dobrze można użyć klasycznej pętli `for`:

```
let linkiStopki = document.querySelectorAll("footer a");
```

```
for (let i = 0; i < linkiStopki.length; i++) {
    linkiStopki[i].style.color = "gray";
}
```

W takiej sytuacji posługujemy się indeksami szuflad tablicy, zamiast nazwą obiektu. Teraz, kiedy rozumiemy już różnicę pomiędzy uchwyceniem jednego elementu albo ich kolekcji, poznajmy inne selektory umożliwiające pracę z wieloma obiektami.

getElementsByTagName() – uchwycenie po znaczniku

Funkcja ta pozwala nam uchwycić wszystkie elementy danego typu (czyli znaczniki tego samego rodzaju) – sama nazwa mówi wszystko: *pobierz elementy według nazwy tagu*. Jest to zatem starsza metoda aniżeli

`querySelectorAll()` – jest oczywiście mniej uniwersalna, bo możemy użyć tylko nazwy tagu, a nie całego silnika selektorów CSS, lecz nadal wykorzystywana. Zwraca kolekcję elementów w postaci obiektu typu `HTMLCollection`. Możemy tą metodą na przykład uchwycić wszystkie pola `input` na stronie:

```
<input type="text" placeholder="Imię">
<input type="text" placeholder="Nazwisko">
<input type="text" placeholder="Miasto">
<button onclick="pokazWszystkieInputy()">Pokaż dane</button>
<p id="wynik"></p>
```

Do ich uchwycenia używamy

`getElementsByTagName("input")` – wszystkie elementy zdefiniowane tym znacznikiem zostaną pobrane do kolekcji, którą możemy wypisać w pętli:

```
function pokazWszystkieInputy() {
    let inputy = document.getElementsByTagName("input");
    let tekst = "Liczba pól input: " + inputy.length + "<br>";
    for (let i = 0; i < inputy.length; i++) {
        tekst += "Zawartość pola numer " + (i + 1) + ": " + inputy[i].value + "<br>";
    }
    document.getElementById("wynik").innerHTML = tekst;
}
```

Oczywiście możemy też dostać się do konkretnego obiektu po prostu używając odpowiedniego indeksu (przy czym numerujemy od zera) – a zatem aby pokazać wartość w polu dla nazwiska (drugie z kolei pole, czyli indeks

`[1]`) moglibyśmy zastosować taką funkcję:

```
function pokazNazwisko() {
    let nazwisko = document.getElementsByTagName("input")[1].value;
    document.getElementById("wynik").innerHTML = nazwisko;
}
```

getElementsByClassName() – uchwycenie po klasie

Jeśli mamy kilka elementów o tej samej klasie i chcemy je wszystkie uchwycić, używamy

`getElementsByClassName()`. Zwraca ona kolekcję `HTMLCollection`, którą można przeglądać za pomocą pętli. Oczywiście dostęp do jednego, wybranego elementu dzięki znajomości jego indeksu również jest możliwy. Załóżmy taką zawartość dokumentu HTML:

```
<input type="text" class="pole" placeholder="Imię">
<input type="text" class="pole" placeholder="Nazwisko">
<input type="number" class="pole" placeholder="Wiek">
<input type="text" placeholder="Miasto"> <!-- ten input nie posiada klasy pole! -->
<button onclick="pokazWszystkie()">Pokaż wartości pól z przypisaną klasę</button>
<button onclick="pokazDrugi()">Pokaż wartość tylko drugiego pola z klasą w
kolekcji</button>
<p id="wynik"></p>
```

Pokażmy w skrypcie JS wartości całej kolekcji albo jedynie w konkretnym, drugim uchwyconym elemencie:

```
function pokazWszystkie() {
    let pola = document.getElementsByClassName("pole");
    let tekst = "";
    // Wypisujemy wszystkie pola z klasy "pole"
    for (let i = 0; i < pola.length; i++) {
        tekst += "Zawartość pola numer " + (i + 1) + ": " + pola[i].value + "<br>";
    }
    document.getElementById("wynik").innerHTML = tekst;
}
function pokazDrugi() {
    let pola = document.getElementsByClassName("pole");
    // Wypisujemy tylko wartość z drugiego inputa z klasą "pole"
    let nazwisko = pola[1].value;
    document.getElementById("wynik").innerHTML = "Nazwisko: " + nazwisko;
}
```

document.forms – uchwyt formularza i jego kontrolek atrybutem name

Kolejny uchwyt – przydatny w pracy z formularzami – pod warunkiem, że jego kontrolki mają przypisane atrybuty `name`:

```
<form name="logowanie">
    <input type="text" name="login" placeholder="Login">
    <input type="password" name="haslo" placeholder="Hasło">
    <button type="button" onclick="pokazDane()">Pokaż dane</button>
</form>
<p id="wynik"></p>
```

Po pobraniu uchwytu formularza, poszczególne wartości możemy pobierać tak jak z tablicy asocjacyjnej (czyli obiekty mają swoje nazwy):

```
function pokazDane() {
    let form = document.forms["logowanie"];
    let login = form["login"].value;
    let haslo = form["haslo"].value;
    document.getElementById("wynik").innerHTML = "Login: " + login + "<br>Hasło: " +
haslo;
}
```

Podsumowanie – który uchwyt wybrać?

- Uchwyt (*ang. reference*) to zmienna (wskaźnik, referencja), która wskazuje konkretny element HTML – tylko mając uchwyt możemy w JS zmienić styl, odczytać dane z pola, ukryć przycisk itd.
- `getElementById()` – najprostszy sposób uchwycenia elementu po atrybucie `id`, np. pola formularza czy akapitu z wynikiem działania skryptu.
- `querySelector()` – pobiera pierwszy pasujący element do selektora CSS – możemy wskazać tag, klasę, `id`, selektor złożony – duża elastyczność! Polecany na egzamin INF.03.
- `querySelectorAll()` – zwraca listę wszystkich pasujących elementów, np. wszystkie linki w stopce – idealne do pracy w pętli.
- `getElementsByTagName()` – uchwyt starszego typu, pobierający elementy tylko po nazwie znacznika (np. wszystkie tagi `<input>`).
- `getElementsByClassName()` – pobiera wszystkie elementy z konkretną klasą – wynik to kolekcja `HTMLCollection`, którą przeglądamy pętlą lub po indeksie.
- `document.forms` – uchwyt do formularza po jego atrybucie `name`, pozwala potem pobierać konkretne pola jako szuflady tablicy asocjacyjnej dla obiektów tego formularza.
- Wszystkie uchwyty typu `getElementsBy...` zwracają kolekcje, a więc możemy je przeglądać pętlą albo używać indeksu celem dostania się do konkretnego elementu.
- `querySelector()` i `querySelectorAll()` działają jak selektory CSS – dlatego są bardzo elastyczne i nowoczesne (wspólnie mogą zastąpić wszystkie inne uchwyty).
- Warto znać różnice pomiędzy uchwytem do jednego elementu a kolekcją wielu elementów – i umieć je przetwarzać.
- Na egzaminie INF.03 często pracujemy z formularzami – poprawne uchwycenie ich elementów to klucz do odczytu danych, sprawdzenia poprawności i dalszego przetwarzania.