Zmiana stylów elementów strony w skryptach

W poprzednich lekcjach mówiliśmy już sporo o tym, jaki sposób uchwycić elementy HTML w dokumencie z użyciem

`getElementById()` czy `querySelector()`. Ponadto, opisaliśmy niuanse tworzenia skryptów przetwarzających liczby (wymagających często parsowania lub konwersji wartości) oraz przetwarzających napisy (które są w JS niemodyfikowalne). Teraz natomiast przejdziemy do trzeciej mechaniki, która równie często pojawia się na egzaminach INF.03 – a mianowicie: jak zmieniać wygląd elementów HTML za pomocą języka JavaScript? Nie mamy tu na myśli modyfikacji arkusza CSS, tylko dynamiczne przypisanie stylów z poziomu skryptu (na przykład po kliknięciu na przycisk).

Dlaczego chcemy zmieniać style z poziomu skryptu?

W trakcie działania witryny bardzo często chcemy coś wizualnie zmienić – np. po kliknięciu przycisku zmienić kolor tła elementu, powiększyć tekst, ukryć jakiś obiekt, nadać mu animację, zmienić margines, wyrównać coś do środka itd. Na przykład realizujemy stronę z quizem online i chcemy zakolorować odpowiednio odpowiedź kliknięta przez użytkownika – zielony kolor oznacza poprawną odpowiedź, czerwony niepoprawną). Albo chcemy aktywować ciemny szablon (skórkę) dla witryny. Wszystko to możemy osiągnąć z poziomu skryptu – wystarczy tylko wiedzieć jak.

Co najważniejsze – każdy odnaleziony w dokumencie element HTML (obiekt) posiada w JS specjalną właściwość o nazwie

`style`. I właśnie ona stanowi dla nas furtkę do zmodyfikowania poszczególnych właściwości CSS. Zobaczmy konkretny przykład – wyobraźmy sobie przycisk, który po kliknięciu zmienia w akapicie kolor tekstu i jego wielkość:

```
<p id="napis">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
<button onclick="zmienStyl()">Zmień styl</button>
```

Dzięki właściwości

`style` zmianę tych konkretnych właściwości CSS zrealizujemy stosunkowo prosto:

```
function zmienStyl() {
    var paragraf = document.getElementById("napis"); //uchwyt akapitu
    // Wykorzystujemy właściwość style aby zmienić konkretne, wybrane wartości w CSS:
    paragraf.style.color = "red";
    paragraf.style.fontSize = "24px";
}
```

Warto wiedzieć, że wszystkie wartości stylów przypisujemy jako napisy (łańcuchy znaków), dlatego nie zapomnijmy o cudzysłowach lub apostrofach przy wartościach – np.

`"blue"`, `"30px"`, `"center"` itd. Powtórzmy też ponownie – nie zmieniamy tutaj zewnętrznego pliku CSS, ale modyfikujemy style konkretnego elementu "w locie" – czyli mówimy przeglądarce: "*Hej, od teraz ten akapit ma mieć kolor czerwony i rozmiar czcionki 24px*" – zostanie to od razu zmienione w hierachii DOM (*ang. Document Object Model*).

Różnica w nazwach właściwości – z myślnika na camelCase

W stylach CSS bardzo często używamy nazw właściwości z myślnikiem – na przykład

`background-color` albo `font-size`. Natomiast JavaScript nie dopuszcza myślnika w nazwach tych samych właściwości – byłby to błąd składniowy, bo przecież operator `-` służy w JS do odejmowania. Dlatego przy zmianie stylów musimy używać tzw. notacji *camelCase* (tłumacząc wprost – notacji "wielbłądziej"), w której każda kolejna część zaczyna się wielką literą (czyli jest to taki "garb" wielbłąda). Oto jak wygląda taka zamiana w praktyce:

Właściwość CSS	Odpowiednik w JavaScript (*camelCase*)
background-color	backgroundColor
font-size	fontSize
text-align	textAlign
margin-top	marginTop
margin-left	marginLeft
padding-right	paddingRight
border-radius	borderRadius
box-shadow	boxShadow
line-height	lineHeight
z-index	zIndex
max-width	maxWidth
min-height	minHeight
overflow-x	overflowX
overflow-y	overflowY
word-spacing	wordSpacing
letter-spacing	letterSpacing
text-transform	textTransform
justify-content	justifyContent
align-items	alignItems
flex-direction	flexDirection
grid-template-	gridTemplateColumns

Właściwość CSS	Odpowiednik w JavaScript (*camelCase*)
columns	
grid-template-rows	gridTemplateRows
animation-duration	animationDuration
animation-timing-function	animationTimingFunction
transition-delay	transitionDelay
transition-property	transitionProperty
border-top-color	borderTopColor

Warunkowa zmiana stylów

Na egzaminie INF.03 może pojawić się zadanie, które wymaga zmiany stylu w zależności od pewnych warunków logicznych. Przykładowo: jeśli liczba wpisana w pole formularza przekracza 100, to tło pola zmieniamy na zielone, w przeciwnym razie na czerwone:

```
function sprawdz() {
    let liczba = document.getElementById("pole").value;
    liczba = parseFloat(liczba);
    let input = document.getElementById("pole");
    if (liczba > 100) {
        input.style.backgroundColor = "lightgreen";
    } else {
        input.style.backgroundColor = "tomato";
    }
}
```

Funkcja

`sprawdz()` może zostać przypisana np. do obsługi zdarzenia keydown lub sprawdzenia wartości po kliknięciu.

Podsumowanie

- Zmiana wyglądu elementów strony w JavaScript to stosunkowo popularne zadanie na egzaminie INF.03 – warto opanować ten mechanizm do perfekcji.
- Do dynamicznej zmiany wyglądu służy właściwość `style` dostępna dla każdego uchwyconego elementu HTML.

- Zmieniając style, nie modyfikujemy zewnętrznego pliku CSS – nadpisujemy style konkretnego elementu “w locie”, czyli bezpośrednio w przeglądarce.
- Wszystkie wartości stylów przypisujemy jako napisy (np. `"blue"`, `"24px"`), nie zapominając o cudzysłowach.
- W JavaScript nie można używać nazw właściwości CSS zawierających myślnik (np. `border-radius`) – musimy je zamienić na notację *camelCase* – polega to w praktyce na tym, iż po pierwszym członie (zapisywanym małymi literami), każdy kolejny wyraz zaczyna się od wielkiej litery – zatem zamiast właściwości `border-radius` znanej z CSS, zapiszemy w JS: `borderRadius`. Przykłady takich modyfikowanych stylów: `fontSize`, `backgroundColor`, `marginTop`, `textAlign`.
- Style CSS możemy zmieniać w skryptach w różnych okolicznościach – np. po kliknięciu przycisku, w odpowiedzi na wpisanie wartości do pola formularza, po załadowaniu strony itp. – decydują wykorzystane przez nas w skryptach zdarzenia (*ang. events*).
- Możemy też zmieniać style warunkowo – np. jeśli użytkownik wpisze wartość większą niż 100, ustawiamy tło pola na zielone, w przeciwnym razie na czerwone.
- Mechanika stylowania przydaje się do tworzenia dynamicznych interfejsów użytkownika – quizów, gier, formularzy walidujących dane itp.
- Dobrą praktyką jest oddzielanie warstwy prezentacyjnej (CSS) od logiki (JavaScript), ale w przypadku szybkich reakcji na działania użytkownika dynamiczna zmiana stylów bywa uzasadniona.
- Zmiana stylów bezpośrednio przez JavaScript to dobry sposób na dodanie interaktywności do naszej strony bez potrzeby przeładowania dokumentu HTML w przeglądarce.

Warto ćwiczyć jak najwięcej – często zmieniajmy style w JS. Tylko tak nauczymy się stosować właściwość `style` i przyzwyczaimy się do notacji *camelCase*. Dzięki temu żadne zadanie egzaminacyjne dotyczące stylizowania obiektów DOM w skryptach JS nie będzie już dla nas zaskoczeniem.