Zadanie 22: DOM – lista zadań

Treść zadania

Wykonaj stronę internetową z listą zadań do wykonania, na której użytkownik dodaje, oznacza i usuwa zadania, wykorzystując edytor zaznaczający składnię oraz przeglądarkę internetową. Utwórz folder o nazwie zadanie22. Po skończonej pracy wszystkie pliki zapisz w tym folderze.

Witryna internetowa

Wygląd strony po otwarciu przedstawia ilustracja 1, a po dodaniu zadań i oznaczeniu części z nich jako wykonane – ilustracja 2.



Ilustracja 1. Strona zadania.html po otwarciu



Ilustracja 2. Strona po dodaniu dwóch zadań i oznaczeniu trzech zadań jako wykonane

Cechy strony zadania.html:

- Zapisana w języku HTML5
- Zadeklarowany polski język zawartości witryny
- Jawnie zastosowany standard kodowania polskich znaków UTF-8
- Tytuł strony „Moje zadania”
- Arkusz stylów w pliku o nazwie styl.css prawidłowo połączony z kodem strony
- Skrypt w pliku o nazwie skrypt.js, połączony ze stroną znacznikiem `<script>` umieszczonym na końcu bloku `<body>`
- Podział strony na bloki: blok główny (`<main>`) umieszczony pośrodku strony i stopka (`<footer>`) pod nim, tak aby układ był zgodny z ilustracją 1
- **Zawartość bloku głównego:** kolejno:
- nagłówek pierwszego stopnia o treści „Moje zadania”
- blok `<div>` o identyfikatorze `dodawanie` zawierający pole tekstowe (`id="nowe"`, tekst podpowiedzi „Wpisz zadanie...”) oraz przycisk (`id="dodaj"`) o treści „Dodaj”
- pusty paragraf o identyfikatorze `komunikat`
- lista nieuporządkowana (`id="lista"`) z trzema elementami: „Zrobić zakupy”, „Napisać skrypt”, „Przeczytać MDN”
- paragraf (`id="stan"`) o treści „Zostało do zrobienia: 3”, w którym liczba 3 jest zapisana w znaczniku `<span>` o identyfikatorze `licznik`

- przycisk (`id="usun"`) o treści „Usuń wykonane”
- **Zawartość bloku stopki:** paragraf o treści „Autor strony: ”, dalej wstawiony numer zdającego, zapisany za pomocą znacznika semantycznego oznaczającego tekst uwypuklony, formatowany domyślnie jako pochylony

Styl CSS witryny internetowej

Styl CSS zdefiniowany jest w całości w zewnętrznym pliku o nazwie styl.css. Cechy formatowania CSS działające na stronie:

- Domyślnie, dla wszystkich selektorów: krój czcionki „Comic Sans MS”, w przypadku braku cursive
- Dla ciała strony: kolor tła Khaki
- Dla bloku głównego: szerokość 500 px, marginesy zewnętrzne górny 40 px, dolny 20 px, lewy i prawy wyliczane automatycznie przez przeglądarkę, marginesy wewnętrzne 30 px, kolor tła LightYellow, obramowanie lewe linią ciągłą o grubości 8 px i kolorze DarkOliveGreen, cień (`box-shadow`) przesunięty o 5 px w prawo i 5 px w dół, rozmycie 10 px, kolor Olive
- Dla nagłówka pierwszego stopnia: górny margines zewnętrzny 0, wyrównanie tekstu do środka, kolor czcionki DarkOliveGreen
- Dla pola tekstowego w bloku o identyfikatorze dodawanie: model pudełkowy border-box, szerokość 75%, marginesy wewnętrzne 8 px, obramowanie linią ciągłą o grubości 2 px i kolorze DarkOliveGreen
- Dla przycisku: marginesy wewnętrzne 8 px (góra i dół) i 14 px (lewy i prawy), kolor tła DarkOliveGreen, kolor czcionki White, brak obramowania, kursor pointer. Gdy kursor znajdzie się na przycisku, kolor tła zmienia się na OliveDrab
- Dla elementu o identyfikatorze komunikat: minimalna wysokość 1.5em, kolor czcionki Crimson
- Dla elementu o identyfikatorze lista: marginesy wewnętrzne 0, brak znaczników listy (`list-style-type: none`)
- Dla elementów `<li>` w elemencie o identyfikatorze lista (selektor potomka): marginesy wewnętrzne 10 px (góra i dół) i 5 px (lewy i prawy), obramowanie dolne linią kropkowaną (`dotted`) o grubości 1 px i kolorze DarkOliveGreen, kursor pointer. Gdy kursor znajdzie się na elemencie, kolor tła zmienia się na Beige
- Dla elementu o identyfikatorze usun: kolor tła Sienna
- Dla stopki: wyrównanie tekstu do środka, kolor czcionki DarkOliveGreen

Uwaga: nie należy zmieniać nazw klas ani identyfikatorów, gdyż jest to uwarunkowane projektem całej witryny.

Skrypt działający po stronie klienta

Wymagania dotyczące skryptu:

- Zapisany w języku JavaScript w pliku skrypt.js, stosuje znaczące nazewnictwo zmiennych i funkcji w języku polskim lub angielskim
- Wszystkie zmiany zachodzą w odpowiedzi na działanie użytkownika

Polecenie 1. Po kliknięciu przycisku „Dodaj” skrypt dodaje na końcu listy nowe zadanie o treści wpisanej w polu tekstowym (bez spacji na początku i na końcu), a następnie czyści pole. Gdy pole jest puste lub zawiera tylko spacje, skrypt nie dodaje zadania i wyświetla w paragrafie komunikat tekst „Wpisz treść zadania.”. Po poprawnym dodaniu zadania paragraf komunikat jest pusty.

Polecenie 2. Naciśnięcie klawisza Enter w polu tekstowym działa tak samo jak kliknięcie przycisku „Dodaj”.

Polecenie 3. Kliknięcie dowolnego zadania z listy (także dodanego przez użytkownika) oznacza je jako wykonane: tekst jest przekreślony (

Polecenie 4. Element o identyfikatorze licznik zawsze pokazuje liczbę zadań niewykonanych. Wartość jest aktualizowana po dodaniu zadania, oznaczeniu zadania i usunięciu zadań.

Polecenie 5. Po kliknięciu przycisku „Usuń wykonane” skrypt usuwa z listy wszystkie zadania oznaczone jako wykonane.

Tabela 1. Wybrane pola i metody modelu DOM języka JavaScript

Wyszukiwanie i zmiana elementów	Dodatkowo dopuszczalne
`document.getElementById(id)`	`element.value, trim()`
`document.getElementsByTagName(TagName)`	`element.style.textDecoration`
`document.createElement(element)`	`e.key (zdarzenie klawiatury)`
`document.appendChild(element)`	`element.parentNode`
`document.removeChild(element)`	`element.length`
`onclick, onkeyup`	`element.style.property_name = "wartość"`

UWAGA: po zakończeniu pracy sprawdź działanie strony w przeglądarce. W folderze powinny znajdować się pliki: zadania.html, styl.css, skrypt.js.
Ocenie podlegać będą 3 rezultaty: zawartość witryny internetowej, styl CSS witryny internetowej, skrypt.

Rozwiązanie

Pytania teoretyczne

Pytanie 1. Dlaczego przy usuwaniu zadań pętlę wykonujemy od końca kolekcji? Co się stanie, gdy wykonamy ją od początku (`for (let i = 0; i < zadania.length; i++)`) i usuniemy wszystkie elementy?

Pytanie 2. Dlaczego dodanie zadania zapisem `lista.innerHTML += "<li>" + tresc + "</li>"` jest gorszym rozwiązaniem niż `createElement()` i `appendChild()`?

Pytanie 3. Do czego służy metoda `trim()` w skrypcie? Co zwróci `"  ".trim()` i jak to wykorzystano?

Pytanie 4. Zmiennej `zadania` przypisano `lista.getElementsByTagName("li")` przy pięciu zadaniach na liście. Ile wyniesie `zadania.length` po usunięciu jednego zadania metodą `removeChild()`?

- **A.** 5
- **B.** 4
- **C.** 0
- **D.** undefined

Pytanie 5. Dlaczego klawisz Enter obsługujemy zdarzeniem keyup i sprawdzamy `e.key`? Którego zdarzenia klawiatury nie należy już używać w nowych projektach?

Odpowiedzi do pytań teoretycznych