Zaawansowany Programista JS

Dzień 4

Plan

Dzien 4 - Programowanie funkcyjne

- Pętle w programowaniu funkcyjnym za pomocą `forEach`
- Znajdowanie elementu za pomocą `find`
- Transformacja tablic za pomocą `map`
- Filtrowanie danych za pomocą `filter`
- Sortowanie danych za pomocą `sort`
- Kalkulacja wyników za pomocą `reduce`
- Tworzenie aplikacji z wykorzystaniem programowania funkcyjnego

---

Pętle w programowaniu funkcyjnym za pomocą `forEach`

Programowanie funkcyjne to paradygmat programowania, który skupia się na definiowaniu funkcji i ich wykorzystywaniu do rozwiązywania problemów. W JavaScript programowanie funkcyjne jest bardzo popularne i często wykorzystywane.

`forEach()` to metoda wbudowana w JavaScript, która pozwala na iterowanie po elementach tablicy i wywoływanie na nich określonej funkcji zwrotnej dla każdego elementu.

Składnia metody forEach() jest następująca:

```
array.forEach((currentValue, index, array) => {
  // kod do wykonania dla każdego elementu
});
```

Parametry metody forEach() to:

`currentValue` - aktualny element tablicy `index` - indeks aktualnego elementu w tablicy `array` - tablica, która jest przetwarzana przez metodę

Przykład użycia metody forEach():

```
let fruits = ['apple', 'banana', 'orange'];

fruits.forEach((fruit, index) => {
```

```
  console.log(`Fruit ${index+1}: ${fruit}`);
});
```

W powyższym przykładzie, metoda `forEach()` jest wykorzystana do iterowania po tablicy `fruits`. Dla każdego elementu tablicy zostaje wywołana funkcja zwrotna, która wyświetla jego wartość i indeks w konsoli.

Metoda `forEach()` jest często wykorzystywana w programowaniu funkcyjnym w JavaScript, ponieważ umożliwia wygodne iterowanie po elementach tablicy i wykonywanie operacji na nich bez potrzeby użycia pętli for lub while.

Znajdowanie elementu za pomocą `find`

`find()` to metoda wbudowana w JavaScript, która służy do znalezienia pierwszego elementu w tablicy, który spełnia warunek określony w funkcji zwrotnej. Metoda `find()` iteruje po elementach tablicy i wywołuje funkcję zwrotną (callback) dla każdego elementu.

Funkcja zwrotna powinna zwrócić wartość logiczną true lub false, w zależności od tego, czy bieżący element spełnia warunek wyszukiwania. Jeśli funkcja zwrotna zwróci wartość true dla danego elementu, to `find()` zwróci ten element. Jeśli nie ma elementów, które spełniają warunek, to `find()` zwróci wartość `undefined`.

Przykład użycia metody find()

```
let numbers = [1, 2, 3, 4, 5];

let even = numbers.find((element) => {
  return element % 2 === 0;
});

console.log(even); // 2
```

W powyższym przykładzie, metoda `find()` jest wykorzystana do znalezienia pierwszego parzystego elementu tablicy numbers. Dla każdego elementu tablicy, funkcja zwrotna sprawdza, czy reszta z dzielenia elementu przez 2 wynosi 0 (czyli czy element jest parzysty). Pierwszy element, dla którego ta wartość wynosi true, zostanie zwrócony przez metodę `find()`.

Transformacja tablic za pomocą `map`

`map()` to metoda wbudowana w JavaScript, która służy do przetwarzania tablicy i tworzenia nowej tablicy z wynikami przetwarzania. Metoda `map()` iteruje po każdym elemencie tablicy, wywołując dla każdego elementu określoną funkcję zwrotną, która zwraca nową wartość. Wyniki przetwarzania są zapisywane w nowej tablicy, która jest zwracana przez metodę `map()`.

Składnia metody `map()` jest następująca:

```
let newArray = array.map((currentValue, index, array) => {
  // kod do wykonania dla każdego elementu
  return result;
});
```

Parametry metody map() to:

`currentValue` - aktualny element tablicy `index` - indeks aktualnego elementu w tablicy `array` - tablica, która jest przetwarzana przez metodę

Funkcja zwrotna przekazywana do metody `map()` przyjmuje trzy parametry: `currentValue`, index i array, a zwraca wartość, która zostanie umieszczona w nowej tablicy.

Przykład użycia metody `map()`

```
let numbers = [1, 2, 3, 4, 5];

let doubledNumbers = numbers.map((number) => {
  return number \* 2;
});


console.log(doubledNumbers); // [2, 4, 6, 8, 10]
```

W powyższym przykładzie, metoda `map()` jest wykorzystana do utworzenia nowej tablicy `doubledNumbers` zawierającej wyniki przemnożenia każdego elementu tablicy numbers przez 2.

Metoda `map()` jest bardzo użyteczna w programowaniu funkcyjnym, ponieważ umożliwia łatwe przetwarzanie tablicy i tworzenie nowej tablicy z wynikami przetwarzania bez potrzeby użycia pętli for lub while.

Filtrowanie danych za pomocą `filter`

`filter()` to metoda wbudowana w JavaScript, która służy do przetwarzania tablicy i filtrowania jej elementów na podstawie określonego warunku. Metoda `filter()` iteruje po każdym elemencie tablicy i wywołuje dla każdego elementu określoną funkcję zwrotną, która zwraca wartość true lub false w zależności od tego, czy dany element powinien zostać zachowany w wynikowej tablicy.

Składnia metody `filter()` jest następująca:

```
let newArray = array.filter((currentValue, index, array) => {
  // kod do wykonania dla każdego elementu
  return result;
});
```

Parametry metody filter() to:

`currentValue` - aktualny element tablicy `index` - indeks aktualnego elementu w tablicy `array` - tablica, która jest przetwarzana przez metodę

Funkcja zwrotna przekazywana do metody `filter()` przyjmuje trzy parametry: `currentValue`, `index` i `array`, a zwraca wartość true lub false w zależności od tego, czy dany element powinien zostać zachowany w wynikowej tablicy.

Przykład użycia metody `filter()`

```
let numbers = [1, 2, 3, 4, 5];
```

```
let evenNumbers = numbers.filter((number) => {
  return number % 2 === 0;
});

console.log(evenNumbers); // [2, 4]
```

W powyższym przykładzie, metoda `filter()` jest wykorzystana do utworzenia nowej tablicy `evenNumbers` zawierającej tylko parzyste liczby z tablicy numbers.

Metoda `filter()` jest bardzo użyteczna w programowaniu funkcyjnym, ponieważ umożliwia łatwe filtrowanie elementów tablicy na podstawie określonych warunków bez potrzeby użycia pętli for lub while.

Sortowanie danych za pomocą `sort`

`sort()` to metoda wbudowana w JavaScript, która służy do sortowania elementów w tablicy. Metoda `sort( )` sortuje elementy w miejscu i zwraca posortowaną tablicę. Domyślnie, sortowanie jest wykonywane w porządku leksykograficznym, czyli według wartości Unicode każdego elementu.

Przykład użycia metody sort():

```
let fruits = ["banana", "apple", "orange", "grape"];

fruits.sort(); // Sortowanie alfabetyczne

console.log(fruits); // ["apple", "banana", "grape", "orange"]
```

W powyższym przykładzie, metoda `sort()` jest wykorzystana do posortowania tablicy fruits w porządku alfabetycznym.

Można również przekazać własną funkcję porównującą do metody `sort()`. Na przykład, jeśli chcemy posortować elementy tablicy numerycznie, można użyć funkcji porównującej w następujący sposób:

```
let numbers = [10, 2, 5, 1, 9];

numbers.sort((a, b) => {
  return a - b;
});

console.log(numbers); // [1, 2, 5, 9, 10]
```

W powyższym przykładzie, przekazana funkcja porównująca zwraca różnicę między dwoma elementami `(a - b)`. Dzięki temu elementy są sortowane numerycznie, a nie leksykograficznie.

Kalkulacja wyników za pomocą `reduce`

`reduce()` to metoda wbudowana w JavaScript, która służy do agregacji (redukowania) wartości w tablicy do jednej wartości. Metoda `reduce()` iteruje po elementach tablicy i wywołuje funkcję zwrotną (callback) dla każdego elementu z dwoma argumentami: wartością akumulatora oraz bieżącym elementem.

Przykład użycia metody `reduce()`

```
let numbers = [1, 2, 3, 4, 5];

let sum = numbers.reduce((acc, val) => {
  return acc + val;
}, 0);

console.log(sum); // 15
```

W powyższym przykładzie, metoda `reduce()` jest wykorzystana do zsumowania wszystkich elementów tablicy numbers. Wartość początkowa akumulatora wynosi zero. Dla każdego elementu tablicy, funkcja zwrotna dodaje wartość elementu do wartości akumulatora. Ostatecznie, wartość zwrócona przez metodę `reduce()` to suma wszystkich elementów tablicy.

Można również użyć metody `reduce()` do innych operacji agregujących, na przykład do znalezienia największej lub najmniejszej wartości w tablicy, albo do utworzenia nowej tablicy z wartościami przetworzonymi przez funkcję zwrotną.

Tworzenie aplikacji z wykorzystaniem programowania funkcyjnego

```
const books = [
  {
    isbn: "9781593279509",
    title: "Eloquent JavaScript, Third Edition",
    subtitle: "A Modern Introduction to Programming",
    author: "Marijn Haverbeke",
    published: "2018-12-04T00:00:00.000Z",
    publisher: "No Starch Press",
    pages: 472,
    description: "JavaScript lies at the heart of almost every modern web application,
from social apps like Twitter to browser-based game frameworks like Phaser and
Babylon. Though simple for beginners to pick up and play with, JavaScript is a
flexible, complex language that you can use to build full-scale applications.",
    website: "http://eloquentjavascript.net/"
  },
  {
    isbn: "9781491943533",
    title: "Practical Modern JavaScript",
    subtitle: "Dive into ES6 and the Future of JavaScript",
    author: "Nicolás Bevacqua",
    published: "2017-07-16T00:00:00.000Z",
    publisher: "O'Reilly Media",
    pages: 334,
    description: "To get the most out of modern JavaScript, you need learn the latest
features of its parent specification, ECMAScript 6 (ES6). This book provides a highly
practical look at ES6, without getting lost in the specification or its implementation
details.",
    website: "https://github.com/mjavascript/practical-modern-javascript"
  },
```

```
  {
    isbn: "9781593277574",
    title: "Understanding ECMAScript 6",
    subtitle: "The Definitive Guide for JavaScript Developers",
    author: "Nicholas C. Zakas",
    published: "2016-09-03T00:00:00.000Z",
    publisher: "No Starch Press",
    pages: 352,
    description: "ECMAScript 6 represents the biggest update to the core of JavaScript in the history of the language. In Understanding ECMAScript 6, expert developer Nicholas C. Zakas provides a complete guide to the object types, syntax, and other exciting changes that ECMAScript 6 brings to JavaScript.",
    website: "https://leanpub.com/understandinges6/read"
  },
  {
    isbn: "9781449365035",
    title: "Speaking JavaScript",
    subtitle: "An In-Depth Guide for Programmers",
    author: "Axel Rauschmayer",
    published: "2014-04-08T00:00:00.000Z",
    publisher: "O'Reilly Media",
    pages: 460,
    description: "Like it or not, JavaScript is everywhere these days -from browser to server to mobile- and now you, too, need to learn the language or dive deeper than you have. This concise book guides you into and through JavaScript, written by a veteran programmer who once found himself in the same position.",
    website: "http://speakingjs.com/"
  },
  {
    isbn: "9781449331818",
    title: "Learning JavaScript Design Patterns",
    subtitle: "A JavaScript and jQuery Developer's Guide",
    author: "Addy Osmani",
    published: "2012-08-30T00:00:00.000Z",
    publisher: "O'Reilly Media",
    pages: 254,
    description: "With Learning JavaScript Design Patterns, you'll learn how to write beautiful, structured, and maintainable JavaScript by applying classical and modern design patterns to the language. If you want to keep your code efficient, more manageable, and up-to-date with the latest best practices, this book is for you.",
    website: "http://www.addyosmani.com/resources/essentialjsdesignpatterns/book/"
  },
  {
    isbn: "9798602477429",
    title: "You Don't Know JS Yet",
    subtitle: "Get Started",
    author: "Kyle Simpson",
    published: "2020-01-28T00:00:00.000Z",
    publisher: "Independently published",
    pages: 143,
    description: "The worldwide best selling You Don't Know JS book series is back for a 2nd edition: You Don't Know JS Yet. All 6 books are brand new, rewritten to cover
```

```
all sides of JS for 2020 and beyond.",
    website: "https://github.com/getify/You-Dont-Know-JS/tree/2nd-ed/get-started"
  },
  {
    isbn: "9781484200766",
    title: "Pro Git",
    subtitle: "Everything you neeed to know about Git",
    author: "Scott Chacon and Ben Straub",
    published: "2014-11-18T00:00:00.000Z",
    publisher: "Apress; 2nd edition",
    pages: 458,
    description: "Pro Git (Second Edition) is your fully-updated guide to Git and its
usage in the modern world. Git has come a long way since it was first developed by
Linus Torvalds for Linux kernel development. It has taken the open source world by
storm since its inception in 2005, and this book teaches you how to use it like a
pro.",
    website: "https://git-scm.com/book/en/v2"
  },
  {
    isbn: "9781484242216",
    title: "Rethinking Productivity in Software Engineering",
    subtitle: "",
    author: "Caitlin Sadowski, Thomas Zimmermann",
    published: "2019-05-11T00:00:00.000Z",
    publisher: "Apress",
    pages: 310,
    description: "Get the most out of this foundational reference and improve the
productivity of your software teams. This open access book collects the wisdom of the
2017 \"Dagstuhl\" seminar on productivity in software engineering, a meeting of
community leaders, who came together with the goal of rethinking traditional
definitions and measures of productivity.",
    website: "https://doi.org/10.1007/978-1-4842-4221-6"
  }
]

// Napisz aplikacje "Moje ksiazki".

// 1. Stworz HTML w ktorym bedzie znajdowal sie formularz z polem input i jednym
przyciskiem (szukaj)
// 2. Stworz HTML z lista ksiazek.
// 3. Zrob obsluge wyswietlania ksiazek ze zmiennej books, zdefiniowanej powyzej.
Wyswietl na ekranie tytul, autor i link do ksiazki
// 4. Zrob obsluge formularza. Po wcisnieciu guzika szukaj, przefiltruj listę wyników,
uzywajac pola title.
// 5*. pod formularzem dodaj select, w ktorym bedzie mozna wybrac po czym chcemy
filtrowac (title albo author)
// 6*. Pod lista ksiazek, stworz formularz "Add Book", ktory dodaje nowa ksiazke do
tablicy. Po dodaniu elementu do tablicy, powinien on móc dać się wyszukać.
```
