Lekcja 13

Temat: Pętle cz. 2. Wybrane obiekty wbudowane JS – cz.2,

Pętle do obsługi obiektów i tablic:

Pętla for…in

Pętla `for..in` w JavaScript umożliwia iterację po wszystkich kluczach właściwości obiektu.

Składnia pętli:

```
for (key in object) {
    // ciało pętli
}
```

W każdej iteracji pętli klucz jest przypisywany do zmiennej key. Pętla jest kontynuowana dla wszystkich właściwości obiektu. Po zdobyciu kluczy łatwo można znaleźć odpowiadające im wartości.

Umożliwia dostęp do każdej właściwości i wartości obiektu bez znajomości konkretnej nazwy właściwości .

Przykład 1. Iteracja przez obiekt

```
<script>
    const student = {
        name: 'Kuba',
        class1: "2c",
        age: 15
    }
    // użycie for...in
    for ( let key in student ) {

        // wyświetlenie właściwości
        console.log(`${key} => ${student[key]}`);
    }
</script>
```

W powyższym programie pętla for...in służy do iteracji obiektu student i wypisania wszystkich jego właściwości.

- Klucz obiektu jest przypisany do zmiennej key.
- student[key] służy do uzyskania dostępu do wartości

Przykład 2.

```
<p id="output1"></p>
<script>
    const output = document.getElementById("output1");
    let contents = ''
    const person = {
        firstName: 'Jan',
        lastName: 'Kowalski',
        birthDay: '01-04-2022'
    };
    for(let prop in person) {
        contents+=`${prop}: ${person[prop]}<br>`;
    }
    output.innerHTML = contents
</script>
```

Przykład 3. **for...in ze stringiem**

Możesz także użyć pętli for...in do iteracji po wartościach ciągu.

```
<script>
    const string = 'ZSK Poznań';

    // użycie pętli for...in
    for (let i in string) {
        console.log(string[i]);
    }

</script>
```

```
Z
S
K

P
o
z
n
a
ń
```

Przykład 4. **for...in z tablicami**

```
<script>
    const arr = [ 'Witaj', 1, 'JavaScript' ];
    for (let x in arr) {
        console.log(arr[x]);
    }

</script>
```

```
Witaj
1
JavaScript
```

Przykład 5. for...in z tablicami rzadkimi – zagrożenia

```
<script>
    let arr = [];
    // ustaw trzeci element na 3, pozostałe elementy są `niezdefiniowane`
    arr[2] = 3;
    console.log('Pętla for:');
    for (let i = 0; i < arr.length; i++) {
        console.log(arr[i]);
    }
    console.log('Pętla for ... in :');
    for (const key in arr) {
        console.log(arr[key]);
    }
</script>
```

```
Pętla for:
undefined
3
Pętla for ... in :
3
```

Po zastosowaniu pętli for…in dane wyjściowe pokazują tylko trzeci element.

UWAGA: Dobrą praktyką jest nieużywanie for...in do iteracji po tablicy , zwłaszcza gdy ważna jest kolejność elementów tablicy. Jednym z lepszych sposobów iteracji po tablicy jest użycie **pętli for...of** lub **metody forEach**

Pętla for...of

Pętla for...of została wprowadzona w późniejszych wersjach JavaScript ES6 .

Pętla for..of w JavaScript pozwala na iterację po iterowalnych obiektach (tablice, zestawy, mapy, łańcuchy itp.).

Składnia pętli:

```
for (element of iterable) {
    // ciało pętli
}
```

Tutaj,

- iterable - obiekt iterowalny (tablica, zestaw, łańcuchy itp.).
- element - elementy w iterowalnym obiekcie, może być zadeklarowany przy pomocy var, let lub const

W każdej iteracji właściwość obiektu iterowalnego jest przypisywana do element

Przykład 6. Pętla for..of może służyć do iteracji po tablicy

```
<script>
    // tablica
    const students = ['Jan', 'Stefan', 'Jacek'];

    // użycie pętli for...of
    for ( let element of students ) {

        // wyświetlenie właściwości
        console.log(element);
    }
</script>
```

Przykład 7. for...of ze stringiem

```
<script>
    const string = 'ZSK Poznań';

    // użycie pętli for...in
    console.log('for...in');
    for (let i in string) {
        console.log(string[i]);
    }
    // użycie pętli for...of
    console.log('for...of');
    for (let i of string) {
        console.log(i);
    }

</script>
```

Jeśli nie zmieniasz zmiennej wewnątrz pętli, powinieneś użyć słowa kluczowego const zamiast słowa kluczowego let w następujący sposób:

```
let scores = [80, 90, 70];


for (const score of scores) {
    console.log(score);
}
```

Aby uzyskać dostęp do indeksu elementów tablicy wewnątrz pętli, możesz użyć instrukcji **`for...of`** z metodą tablicy **`entries()`.**

Metoda `array.entries()` zwraca parę `[index, element]` w każdej iteracji. Na przykład:

Przykład 8.

```
let colors = ['Red', 'Green', 'Blue'];

for (const [index, color] of colors.entries()) {
    console.log(`${color} is at index ${index}`);
}
```

Wyjście:

```
Red is at index 0
Green is at index 1
Blue is at index 2
```

Pętla for...of nie może być używana do iteracji po obiekcie!!

Pętla `for...of` została wprowadzona w ES6 . Niektóre przeglądarki mogą nie obsługiwać jego użycia. Nie jest obsługiwana w programie Internet Explorer.

Obiekt Array

Metody JavaScript Array umożliwiają efektywne manipulowanie tablicami

Metoda forEach()

Metoda forEach() wywołuje funkcję i iteruje po elementach **tablicy**.

Metodę forEach() można również stosować na mapach i zestawach.

Składnia metody:

```
array.forEach(function(currentValue, index, arr))
```

Tutaj,

- `function(currentValue, index, arr)` - funkcja do uruchomienia dla każdego elementu tablicy
- `currentValue` - wartość tablicy

- `index` (opcjonalnie) - indeks bieżącego elementu
- `arr` (opcjonalnie) - tablica bieżących elementów

Przykład 9.

```
<script>
    const students = ['Jan', 'Stefan', 'Jacek'];
    // użycie pętli for...of
    console.log("for ...of:")
    for ( let element of students ) {
        console.log(element);
    }
    // użycie metody forEach
    console.log("forEach:")
    students.forEach(function (e) {
        console.log(e);
    });
</script>
```

Przykład 10. **Aktualizacja elementów tablicy**

```
<script>
    const students = ['Jan', 'Stefan', 'Jacek'];

    console.log("forEach:")
    students.forEach(function (e) {
        console.log(e);
    });
    //Aktualizacja elementów tablicy
    console.log("Aktualizacja:")
    students.forEach(myFunction);
    function myFunction(item, index, arr) {
        arr[index] = 'Cześć ' + item;
    }

    console.log(students);
</script>
```

Przykład 11. **forEach() z funkcją strzałkową**

```
<script>
    const students = ['Jan', 'Stefan', 'Jacek'];
    console.log("zwykła funkcja w forEach:")
    students.forEach(function (e) {
        console.log(e);
    });
    //użycie funkcji strzałkowej
    console.log("funkcja strzałkowa w forEach:");
    students.forEach(element => {
        console.log(element);
    });
</script>
```

Jednym z ograniczeń metody forEach()w porównaniu z pętlą for jest to, że nie można użyć instrukcji break lub continue do sterowania pętlą.

Właściwość length

Z definicji właściwość **length** tablicy jest 32-bitową liczbą całkowitą bez znaku, która jest zawsze numerycznie większa niż najwyższy indeks w tablicy.

Wartość długości wynosi 2^{32} . Oznacza to, że tablica może pomieścić do 4294967296 (2^{32}) elementów.

Właściwość **length** zachowuje się różnie w zależności od typów tablic, w tym gęstych i rzadkich.

1) **Gęste tablice**

Gęsta tablica to tablica, w której jej elementy mają ciągłe indeksy zaczynające się od zera.

W przypadku gęstych tablic można użyć właściwości **length**, aby uzyskać liczbę elementów w tablicy.

2) **Rzadkie tablice**

Rozrzedzona tablica to tablica, której elementy nie mają ciągłych indeksów zaczynających się od zera.

Na przykład [10,, 20, 30] to tablica rzadka, ponieważ indeksy jej elementów to 0, 2 i 3.

W tablicy rzadkiej właściwość **length** nie wskazuje rzeczywistej liczby elementów. Jest to liczba większa niż najwyższy indeks.

Przykład 12:

```
<h3>Obiekt - Array</h3>
<p>efekt w konsoli</p>
<script>
    //tablica gęsta:
    let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];
    console.log(mountains);
    console.log(mountains.length); // 3

    //tablica rzadka:
    let numbers = [10, , 20, 30];
    console.log(numbers);
    console.log(numbers.length); // 4
    //dodanie elementu do  tablicy numbers o indeksie 10
    //numbers[10] = 100;
    console.log(numbers);
    console.log(numbers.length); // 11
</script>
```

Modyfikowanie właściwości długości tablicy JavaScript

JavaScript umożliwia zmianę wartości właściwości length tablicy. Zmieniając wartość długości, możesz usunąć elementy z tablicy lub rozrzedzić tablicę.

1) Opróżnienie tablicy: Jeśli ustawisz długość na zero, tablica będzie pusta :

2) Usuwanie elementów: Jeśli ustawisz właściwość length tablicy na wartość mniejszą niż najwyższy indeks, wszystkie elementy, których indeks jest większy lub równy nowej długości, zostaną usunięte.

3) Stworzenie tablicy rzadkiej: Jeśli ustawisz właściwość length tablicy na wartość wyższą niż najwyższy indeks, tablica będzie rzadka:

Przykład 13:

```
<h3>Obiekt - Array</h3>
<p>efekt w konsoli</p>
<script>
    //Opróżnij tablicę
    let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];
    mountains.length = 0;
    console.log(mountains);
    console.log(mountains.length);

    //Usuwanie elementów
    const fruits = ['Apple', 'Orange', 'Strawberry', 'Lemon'];
    fruits.length = 2;
    console.log(fruits);

    // przekształcenie do tablic rzadkiej
    const fruits2 = ['Apple', 'Orange', 'Strawberry'];
    fruits2.length = 5;
    console.log(fruits2);
</script>
```

Dodawanie/usuwanie elementów

- **`push()`** – dodaje jeden lub więcej elementów na końcu tablicy.
- **`unshift()`** – dodaje jeden lub więcej elementów na początku tablicy.
- **`pop()`** – usuwa element z końca tablicy.
- **`shift()`** – usuwa pierwszy element z tablicy.
- **`splice()`** – manipuluje elementami w tablicy, takimi jak usuwanie, wstawianie i zastępowanie elementów.
- **`slice()`** – kopiuje elementy tablicy.

`push()` Dodanie elementu na końcu tablicy

Metoda Array.prototype.push() dodaje jeden lub więcej elementów na końcu tablicy i zwraca długość nowej tablicy. (**Metoda push**() zwraca nową wartość właściwości length obiektu tablicy, na którym wywołujesz metodę.)

Przykład14:

```
<h3>Obiekt - Array</h3>
<p>efekt w konsoli</p>
<script>
    //Używanie array push() do dołączenia jednego elementu do tablicy
    let numbers = [10, 20, 30];
    const length = numbers.push(40);
    console.log(length);
    console.log(numbers);

    //Użycie array push() do dodania wielu elementów na końcu tablicy
    const fruits = ['Apple', 'Orange'];
    const lengthF = fruits.push( 'Strawberry', 'Lemon');
    console.log(lengthF);
    console.log(fruits);
</script>
```

Przykład15: Używanie push() do dołączania elementów tablicy do innej tablicy; Począwszy od wersji ES6, można użyć też operatora spread (...)

```
<h3>Obiekt - Array</h3>
<p>efekt w konsoli</p>
<script>
    // Używanie push() do dołączania elementów tablicy do innej
    tablicy
    let colors = ['red', 'green', 'blue'];
    let cmyk = ['cyan', 'magenta', 'yellow', 'back'];
    // użycie pętli for...of
    for (const color of cmyk) {
      colors.push(color);
    }
    console.log(colors);

    // Używanie push() do dołączania elementów tablicy do innej
    tablicy
    let colors1 = ['red', 'green', 'blue'];
    let cmyk1 = ['cyan', 'magenta', 'yellow', 'back'];
    //użycie operatora rozsunięcia ( ...),
    colors1.push(...cmyk1);
    console.log(colors1);
</script>
```

Operator spread jest oznaczony trzema kropkami (…).
Operator spread rozpakowuje elementy obiektów iterowalnych, takich jak tablice, zestawy i mapy, do listy.
Operatora spread można użyć do klonowania obiektu iterowalnego lub łączenia obiektów iterowalnych w jeden.

Więcej: https://www.javascripttutorial.net/es6/javascript-spread/

Przykład 16. pętla for, forEach() i push

```
<script>
    const arrayItems = ['item1', 'item2', 'item3'];
    const copyItems = [];
    const copyItems2 = [];
    //  for
    console.log('pętla for:');
    for (let i = 0; i < arrayItems.length; i++) {
      copyItems.push(arrayItems[i]);
    }
    console.log(copyItems);
    //  forEach
    console.log('pętla forEach:');
    arrayItems.forEach(function(item){
      copyItems2.push(item);
    })
    console.log(copyItems2);
</script>
```

`unshift()` – dodanie elementów na początku tablicy.

Metoda Array.prototype.unshift() dodaje jeden lub więcej elementów na początku tablicy i zwraca długość nowej tablicy .

Metoda wymaga ponownego indeksowania istniejących elementów, jest powolna, jeśli tablica zawiera wiele elementów.

Przykład 17.

```
<script>
    //Używanie array unshift() do dołączenia jednego elementu do tablicy
    let numbers = [10, 20, 30];
    const length = numbers.unshift(40);
    console.log(length);
    console.log(numbers);

    //Użycie array unshift() do dodania wielu elementów
    const fruits = ['Apple', 'Orange'];
    const lengthF = fruits.unshift( 'Strawberry', 'Lemon');
    console.log(lengthF);
    console.log(fruits);
</script>
```

pop() - Usuwanie elementu z końca tablicy

Metoda Array.prototype.pop()usuwa ostatni element z tablicy i zwraca usunięty element.

Metoda pop()zmienia właściwość length tablicy. Jeśli tablica jest pusta, pop()zwraca undefined.

Przykład 18.

```
<script>
    //Używanie array pop() do usunięcia jednego elementu z końca tablicy
    const numbers = [10, 20, 30];
    const last = numbers.pop();
    console.log(last);
    console.log(numbers.length);

    //Używanie array pop() z pustą tablicą
    const fruit = [];
    const last1 = fruit.pop();
    console.log(last1);
    console.log(fruit.length);
</script>
```

shift() - Usuwanie elementu z początku tablicy

Usuwa pierwszy element z tablicy i zwraca ten element. Metoda shift()musi ponownie zindeksować wszystkie pozostałe elementy tablicy. Dlatego jest wolniejsza w porównaniu z metodą pop().

Przykład 19.

```
<script>
    //Używanie array shift() do usunięcia jednego elementu z początku tablicy
    const numbers = [10, 20, 30];
    let number = numbers.shift();
    console.log(number );
    console.log(numbers );
    console.log(numbers.length);
    //użycie metodyshift() z pętlą while , aby usunąć wszystkie elementy tablicy
    const fruit = ["jabłko", "śliwka", "gruszka"];
    while ((fruit1 = fruit.shift()) != undefined) {
      console.log(fruit1);
    }
    console.log(fruit.length);
</script>
```

splice()- usuwanie istniejących elementów, wstawianie nowych elementów i zastępowanie elementów w tablicy.

Pozwala wstawiać nowe elementy w środku tablicy. Możesz jednak użyć tej metody, aby usunąć i zastąpić istniejące elementy.

Metoda splice() zmienia oryginalną tablicę i zwraca tablicę zawierającą usunięte elementy.

Usuwanie elementów za pomocą splice()

```
Array.splice(position,num);
```

position określa pozycję pierwszego elementu do usunięcia, a argument num określa liczbę elementów do usunięcia.

Przykład 20.

```
<script>
    //Usuwanie elementów za pomocą splice()
    const fruit = ["jabłko", "śliwka", "gruszka","truskawka"];
    let deletedFruit = fruit.splice(1, 2);
    console.log(fruit);
    console.log(deletedFruit);
</script>
```

Wstawianie elementów metodą splice()

Można wstawić jeden lub więcej elementów do tablicy, przekazując trzy lub więcej argumentów do metody splice()z drugim argumentem równym zero.

```
Array.splice(position,0,new_element_1,new_element_2,...);
```

- Drugi argument to zero (0), który instruuje metodę splice(), aby nie usuwała żadnych elementów tablicy.
- Trzeci argument, czwarty argument itd. to nowe elementy wstawiane do tablicy.

W tym przypadku metoda nie usuwa żadnych elementów, dlatego zwraca pustą tablicę.

Przykład 21.

```
<script>
    //Wstawianie elementów za pomocą splice()
    const fruit = ["jabłko", "śliwka", "gruszka","truskawka"];
    let deletedFruit = fruit.splice(2, 0, 'pomarańcza', 'kiwi');
    console.log(fruit);
    console.log(deletedFruit);
</script>
```

Podmienianie elementów metodą splice()

Metoda splice() pozwala na wstawianie nowych elementów do tablicy przy równoczesnym usuwaniu istniejących elementów.

Aby to zrobić, przekazujesz co najmniej trzy argumenty z drugim, który określa liczbę elementów do usunięcia, a trzecim, który wskazuje elementy do wstawienia.

Zauważ, że liczba elementów do usunięcia nie musi być taka sama jak liczba elementów do wstawienia.

Przykład 22.

```
<script>
    //Podmienianie elementów za pomocą splice()
    const fruit = ["jabłko", "śliwka", "gruszka","truskawka"];
    let deletedFruit = fruit.splice(1, 2, 'pomarańcza', 'kiwi');
    console.log(fruit);
    console.log(deletedFruit);

    let languages = ['C', 'C++', 'Java', 'JavaScript'];
    languages.splice(2,1,'C#','Swift','Go');
    console.log(languages);
</script>
```