Obiekty w języku JavaScript

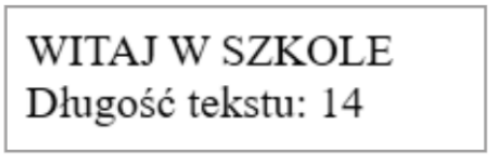
Język JavaScript jako język zorientowany obiektowo udostępnia wiele wbudowanych obiektów. Daje również możliwość odczytywania ich właściwości oraz korzystania z ich metod. Właściwości obiektu reprezentują jego cechy (na przykład liczbę znaków łańcucha, rozmiary okna) lub pozwalają określić jego stan. Nazywane są również polami obiektu, zmiennymi lub zmiennymi składowymi. Metody to funkcje, które wykonują różne operacje na obiekcie.

Do właściwości i metod można się odwołać podobnie jak do zwykłych zmiennych i funkcji, trzeba tylko przed ich nazwą umieścić nazwę obiektu, którego są elementami (na przykład nazwę zmiennej, która przechowuje dany obiekt), i kropkę.

Przykład 3.42

```
var napisz = "Witaj w szkole";
document.write(napisz.toUpperCase() + "<br>");
document.write("Długość tekstu: " + napisz.length);
```

W podanym przykładzie zmienna `napisz` przechowuje obiekt. Funkcja `toUpperCase()` jest metodą obiektu, a `napisz.length` jego właściwością. Wynikiem będzie wypisanie tekstu pokazanego na rysunku 3.6.



Rysunek 3.6. Zastosowanie metody obiektu i jego właściwości

Każdy element strony internetowej jest traktowany jako obiekt związany z obiektem `document`. Obiekty języka JavaScript zawierają informacje opisujące stronę i jej środowisko.

3.6.1. Tworzenie obiektów

Obiekty w języku JavaScript można tworzyć:

- używając literału obiektu,
- za pomocą konstruktora obiektu i słowa kluczowego `new`.

Tworzenie obiektu z użyciem literału

Aby utworzyć nowy obiekt z użyciem literału, należy skorzystać z konstrukcji, która zdefiniuje nazwę obiektu oraz pozwoli utworzyć jego właściwości i metody.

Ponieważ obiekty są zmiennymi, które mogą zawierać wiele wartości, można utworzyć zmienną, która jest obiektem. Każda jej wartość będzie właściwością obiektu, zostanie zapisana jako para nazwa i wartość (*nazwa: wartość*) i umieszczona w nawiasach

klamrowych {}. Działania, jakie można wykonywać na takim obiekcie, zostaną zdefiniowane jako metody i zapisane jako definicje funkcji.

Przykład 3.43

```
var osoba = {
  nazwisko: "Nowacki",
  imie: "Marek",
  zawod: "informatyk",
  pokaz: function() {
    document.write(this.nazwisko + ' ' + this.imie);
  }
};
```

Utworzony obiekt ma trzy właściwości (`nazwisko`, `imie`, `zawod`) i jedną metodę, `pokaz`, która wyświetla nazwisko i imię. Metoda jest funkcją zdefiniowaną w obrębie obiektu. Przy definiowaniu obiektu deklarowane właściwości i funkcje muszą być oddzielone przecinkami. Słowo kluczowe `this` pozwala odwołać się do właściwości lub metod danego obiektu z jego wnętrza. W tym wypadku metoda `pokaz` odwołuje się do właściwości obiektu `osoba`.

Dostęp do właściwości obiektu uzyskamy po podaniu nazwy obiektu i jego właściwości oddzielonych znakiem kropki (na przykład `osoba.nazwisko`). W ten sam sposób można wywołać metodę (`osoba.pokaz`).

Przykład 3.44

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Obiekt osoba</title>
</head>
<body>
<script>
var osoba = {
  nazwisko: "Nowacki",
  imie: "Marek",
  zawod: "informatyk",
```

```
  pokaz: function() {
    document.write(this.nazwisko + ' ' + this.imie);
  }
};
osoba.pokaz();
</script>
</body>
</html>
```

W podanym przykładzie po utworzeniu obiektu `osoba` za pomocą konstrukcji `osoba.pokaz()` została wywołana metoda, która wyświetli nazwisko i imię osoby.

Dla istniejących obiektów można deklarować nowe właściwości i metody.

Przykład 3.45

```
var osoba = {
  nazwisko: "Nowacki",
  imie: "Marek",
  zawod: "informatyk",
  pokaz: function() {
    document.write(this.nazwisko + ' ' + this.imie);
  }
};
osoba.wiek = 19;
osoba.wypisz_wiek = function() {
  document.write('Wiek: ' + this.wiek + ' lat');
}
osoba.wypisz_wiek();
```

W podanym przykładzie po utworzeniu obiektu `osoba` i zdefiniowaniu jego właściwości i metod zostały zadeklarowane nowa właściwość `wiek` oraz nowa metoda `wypisz_wiek()`. W kolejnym kroku została wywołana nowa metoda `osoba.wypisz_wiek()`, za pomocą której został wyświetlony wiek osoby.

Ćwiczenie 3.12

Utwórz w języku JavaScript obiekt opisujący pojazd. Zdefiniuj dla niego właściwości: `marka`, `model`, `rok_produkcji`, `przebieg` oraz metodę `wyswietlDane()` do wyświetlenia marki, modelu i roku produkcji. Wywołaj metodę `wyswietlDane()`, a następnie dodaj właściwość `numer_rej` i metodę `wyswietlDetal()` do wyświetlenia marki samochodu i numeru rejestracyjnego.

Rozwiązanie

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Obiekt - literał</title>
</head>
<body>
<h2>Ćwiczenie - Tworzenie obiektu za pomocą literału</h2>
<script>
var pojazd = {
  marka: "Ford",
  model: "Mustang",
  rok_produkcji: "2020",
  przebieg: 15000,
  wyswietlDane: function() {
    document.write(this.marka + ' ' + this.model + ', rok produkcji: '
+ this.rok_produkcji + "<br>");
  }
};
pojazd.wyswietlDane();

pojazd.numer_rej = "WA34789";
pojazd.wyswietlDetal = function() {
  document.write('Marka: ' + this.marka + ', numer rejestracyjny: '
+ this.numer_rej);
}
pojazd.wyswietlDetal();
```

```
pojazd.numer_rej = "WA34789";
pojazd.wyswietlDetal = function() {
  document.write('Marka: ' + this.marka + ', numer rejestracyjny: '
+ this.numer_rej);
}
pojazd.wyswietlDetal();
</script>
</body>
</html>
```

3.6.2. Tworzenie obiektów z użyciem konstruktora

W języku JavaScript istnieje możliwość tworzenia wielu obiektów mających podobne właściwości. W tym celu można posłużyć się konstruktorem obiektu. Konstruktor przypomina zwykłą funkcję.

Przykład 3.46

```
function Klient(nazwisko_k, imie_k, zawod_k) {
  this.nazwisko = nazwisko_k;
  this.imie = imie_k;
  this.zawod = zawod_k;
  this.wypisz = function() {
    alert(this.nazwisko + ' ' + this.imie);
  }
}
```

Został utworzony konstruktor o nazwie `Klient` z właściwościami `nazwisko`, `imie`, `zawod` oraz metodą `wypisz()`. Właściwościom obiektu zostały przypisane wartości parametrów. Użyte słowo kluczowe `this` odnosi się do aktualnego obiektu i pozwala na przypisanie wartości parametru do odpowiedniego pola tego obiektu.

Słowo kluczowe new

Do utworzenia nowego obiektu na podstawie konstruktora stosowane jest słowo kluczowe `new`.

Przykład 3.47

```
var osoba1 = new Klient('Kowalski', 'Jan', 'kierowca');
var osoba2 = new Klient('Nowak', 'Anna', 'sekretarka');
```

Powstały dwa nowe obiekty `osoba1` i `osoba2` należące do klasy `Klient`.

Właściwości obiektu istnieją w porządku, w jakim zostały zdefiniowane. Można się do nich odwoływać na dwa sposoby:

nazwa_obiektu.nazwa_właściwości

lub

nazwa_obiektu["*nazwa_właściwości*"]

Przykład 3.48

```
osoba1.nazwisko;
osoba1.["nazwisko"];
```

Ćwiczenie 3.13

Utwórz konstruktor obiektu `Uczen` z właściwościami: `nazwisko`, `imie`, `wiek`, `klasa`, `sport` oraz metodę `wypisz()`, która wyświetli właściwości: `nazwisko`, `imie`, `wiek` oraz `sport`. Na podstawie konstruktora utwórz pięć obiektów (uczniów), które będą zawierały `nazwisko`, `imie`, `wiek`, `klasa`, `sport`. Przy tworzeniu obiektów posłuż się zaprojektowanym konstruktorem obiektu. Dla każdego ucznia wyświetl jego nazwisko, imię, wiek oraz sport, jaki uprawia.

Rozwiązanie

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Konstruktor</title>

</head>

<body>

<h2>Ćwiczenie - Konstruktor</h2>

<script>

function Uczen(nazwisko, imie, wiek, klasa, sport) {

  this.nazwisko_u = nazwisko;

  this.imie_u = imie;

  this.wiek_u = wiek;

  this.klasa_u = klasa;
```

```
  this.sport_u = sport;
  this.wypisz = function() {
    document.write(this.nazwisko_u + ' ' + this.imie_u + ' '
+ this.wiek_u + ' lat, sport: ' + this.sport_u + '.' + "<br>");
  }
}

var uczen1 = new Uczen("Nowak", "Paweł", 12, "5b", "tenis");
var uczen2 = new Uczen("Polak", "Anna", 13, "6a", "pływanie");
var uczen3 = new Uczen("Czaja", "Maciej", 13, "6a", "siatkówka");
var uczen4 = new Uczen("Malak", "Julia", 12, "5c", "gimnastyka");
var uczen5 = new Uczen("Wojak", "Michał", 13, "6b", "pływanie");
uczen1.wypisz();
uczen2.wypisz();
uczen3.wypisz();
uczen4.wypisz();
uczen5.wypisz();

</script>
</body>
</html>
```

Właściwość prototype

Nie można dodawać nowych właściwości i metod do konstruktora obiektów tak, jak dodaje się właściwości i metody do istniejącego obiektu. Nowe właściwości i metody muszą być umieszczane w definicji konstruktora.

Sposobem na deklarowanie nowych metod i właściwości dla konstruktora jest wykorzystanie właściwości `prototype`.

Przykład 3.49

```
function Klient(nazwisko, imie) {
  this.nazwisko_k = nazwisko;
  this.imie_k = imie;
}
Klient.prototype.zawod = 'kierowca';
Klient.prototype.pisz_dane = function() {
  document.write(this.nazwisko_k + ' ' + this.imie_k);
}
var osoba1 = new Klient("Malinowski", "Oskar", "kierowca");
osoba1.pisz_dane();
```

W definicji konstruktora zostały zadeklarowane dwie właściwości. Po użyciu właściwości `prototype` została dodana metoda `pisz_dane()` oraz właściwość `zawod`. Od tej pory każdy obiekt, który będzie tworzony na podstawie konstruktora `Klient`, będzie miał tę dodatkową właściwość i metodę.

Właściwość `prototype` może być również wykorzystana do dołączania dodatkowych metod lub właściwości do istniejących obiektów.

Ćwiczenie 3.14

Uzupełnij kod utworzony w ćwiczeniu 3.13. Zmień imię ucznia `Nowak` z `Paweł` na `Karol` i klasę uczennicy `Malak` z `5c` na `5a`. Do konstruktora dodaj metodę `wypisz-Klasa()`, która wyświetli nazwisko, imię ucznia i klasę. Dla każdego ucznia wywołaj utworzoną metodę.

3.7. Obiekty wbudowane języka JavaScript

Niektóre predefiniowane obiekty języka JavaScript to:

- `String` — łańcuch tekstowy,
- `Array` — tablica,
- `Date` — data,
- `Math` — obiekt do przeprowadzania operacji matematycznych.

3.7.1. Obiekt String

Obiektem zawsze występującym w języku JavaScript jest obiekt `String`. Ma on jedną właściwość, `length`, określającą długość łańcucha.

Przykład 3.50

```
var tekst = "Obiekty języka JavaScript";
var dlug = tekst.length;
```

Zmiennej `dlug` przypisana zostanie wartość `25` określająca długość tekstu.

Obiekt `String` ma dwa typy metod.

Pierwszy odnosi się do utworzonego łańcucha, przykładem jest metoda `substring()`, która zwraca podzbiór tego łańcucha. Jej parametry określają położenie początku i końca podzbioru.

Przykład 3.51

```
var tekst = "Obiekty języka JavaScript";
var x = tekst.substring(15, 19);
```

Zostanie zwrócony łańcuch `Java`.

Metodami, które można wykorzystać do zmiany wielkości liter, są: `toUpperCase()` i `toLowerCase()`. Metoda `toUpperCase()` zamienia wszystkie litery ciągu na duże, a metoda `toLowerCase()` zamienia wszystkie litery ciągu na małe.

Przykład 3.52

```
var tekst = "Obiekty języka JavaScript";
var x = tekst.toLowerCase();
```

Zostanie zwrócony łańcuch `obiekty języka javascript`.

Przykład 3.53

```
var tekst = "Obiekty języka JavaScript";
var x = tekst.toUpperCase();
```

Zostanie zwrócony łańcuch `OBIEKTY JĘZYKA JAVASCRIPT`.

Do istniejących obiektów można dodawać nowe właściwości i można definiować dla nich nowe metody.

Do obiektu `String` dołączymy dodatkową funkcję, której wywołanie spowoduje, że pierwsza litera łańcucha zostanie zmieniona na dużą.

Przykład 3.54

```
String.prototype.duzaLitera = function() {
  return this.charAt(0).toUpperCase() + this.substr(1);
}
```

Metoda `charAt()` zwraca znak z pierwszej pozycji łańcucha znaków. Natomiast metoda `substr()` zwraca podzbiór łańcucha znaków. Jako parametr tej metody został podany indeks pierwszego znaku podzbioru. Zadeklarowana metoda została dołączona do obiektu `String` i od tej pory każdy nowy tekst będzie miał metodę, która zamieni jego pierwszą literę na dużą.

Przykład 3.55

```
var tx1 = 'komputer';
document.write(tx1.duzaLitera());
```

W wyniku wykonania skryptu wyświetli się tekst `Komputer`.

3.7.2. Obiekt Array

Tablice służą do przechowywania wielu zmiennych. W języku JavaScript do pracy z tablicami można używać wbudowanego obiektu `Array`. Ma on metody do manipulowania tablicami zmiennych.

Aby utworzyć nową tablicę, należy zadeklarować obiekt `Array` w postaci:

```
var nazwaTablicy = new Array();
```

lub

```
var nazwaTablicy = [];
```

Jeżeli w nawiasach zostanie podana liczba `n`, to zostanie utworzona tablica zawierająca `n` pustych elementów.

Przykład 3.56

```
var tab1 = new Array(10);
var tab2 = [15];
```

W pierwszym przypadku uzyskamy tablicę zawierającą 10 pustych elementów, w drugim powstanie tablica zawierająca jeden element o wartości 15.

Tablicę można również tworzyć, wstawiając do niej konkretne wartości.

Przykład 3.57

```
var tab3 = new Array('Anna', 'Adam', 'Piotr', 'Ewa');
var tab4 = ['Paweł', 'Marcin', 'Ela'];
```

Żeby uzyskać dostęp do elementów tablicy, należy podać numer indeksu danego elementu. Elementy są indeksowane od zera.

Przykład 3.58

```
document.write(tab3[2]);
```

W wyniku wyświetli się wartość `Piotr`.

Aby dodać nową wartość do tablicy, należy przypisać tę wartość do odpowiedniego indeksu tablicy.

Przykład 3.59

```
var tab5 = ['kot', 'pies', 'koń'];
tab5[3] = 'mysz';
tab5[4] = 'chomik';
document.write(tab5[0] + ' i ' + tab5[3]);
```

W wyniku wyświetli się tekst `kot i mysz`.

Dzięki właściwości `length` można określić, z ilu elementów składa się tablica. Jest to bardzo przydatna właściwość, szczególnie gdy chcemy utworzyć pętlę odczytującą wszystkie elementy tablicy.

Przykład 3.60

```
var imie = ['Anna', 'Adam', 'Piotr', 'Ewa', 'Paweł', 'Marcin', 'Ela'];
for (var i = 0; i < imie.length; i++) {
  document.write(imie[i] + "<br>");
}
```

W wyniku zostaną wyświetlone po kolei wszystkie elementy tablicy.

Zadanie 3.1

Zmień zapis skryptu podanego w przykładzie 3.60, tak aby wyświetlone zostały elementy tabeli od ostatniego do pierwszego. Pamiętaj, że indeksowanie elementów rozpoczyna się od zera.

Tablice wielowymiarowe

W języku JavaScript można również tworzyć tablice wielowymiarowe. Wtedy element tablicy jest opisywany za pomocą indeksu określającego jego położenie w wierszu i kolumnie.

Przykład 3.61

```
var Osoby = [];
Osoby[0] = ['Anna', 'Nowak'];
Osoby[1] = ['Adam', 'Kowal'];
Osoby[2] = ['Piotr', 'Ogórek'];
Osoby[3] = ['Ewa', 'Lisowska'];
document.write('imię: ' + Osoby[0][0] + ', nazwisko: ' + Osoby[0][1]
+ "<br>");
document.write('imię: ' + Osoby[1][0] + ', nazwisko: ' + Osoby[1][1]
+ "<br>");
document.write('imię: ' + Osoby[2][0] + ', nazwisko: ' + Osoby[2][1]
+ "<br>");
document.write('imię: ' + Osoby[3][0] + ', nazwisko: ' + Osoby[3][1]);
```

Łączenie elementów tablicy

Za pomocą metody `join()` można łączyć elementy tablicy w jeden ciąg znaków. W metodzie tej można opcjonalnie podać parametr, który określi znak oddzielający kolejne elementy tablicy. Jeżeli nie zostanie podana wartość tego parametru, domyślnym znakiem będzie przecinek.

Przykład 3.62

```
var Tablica = new Array('Anna', 'Adam', 'Piotr');
document.write(Tablica.join() + "<br>");
document.write(Tablica.join(" - ") + "<br>");
```

Odwracanie kolejności elementów tablicy

Za pomocą metody `reverse()` można odwrócić kolejność elementów tablicy.

Przykład 3.63

```
var Tablica = new Array('Anna', 'Adam', 'Piotr');
document.write(Tablica.join() + "<br>");
Tablica.reverse()
document.write(Tablica.join() + "<br>");
```

Sortowanie

Do sortowania elementów tablicy służy metoda `sort()`.

Przykład 3.64

```
var Tablica = new Array('Paweł', 'Anna', 'Maria', 'Adam', 'Piotr');
Tablica.sort();
document.write(Tablica.join());
```

Przykład 3.65

```
var Tablica = new Array(3000, 4567, 12459, 406745);
Tablica.sort();
document.write(Tablica.join());
```

Domyślnie tablica jest sortowana leksykograficznie. Powoduje to, że liczba `12459` będzie mniejsza od `4567`, ponieważ w liczbie `12459` cyfra na pierwszej pozycji jest mniejsza. Aby to zmienić, można sortować tablicę według własnych kryteriów. Należy skorzystać z dodatkowego parametru metody `sort()`. Parametrem będzie własna funkcja sortująca. Tworząc taką funkcję, należy pamiętać o trzech zasadach:

- jeżeli `funkcja(a, b)` zwróci wartość mniejszą od `0`, to wartości `a` zostanie nadany indeks mniejszy od indeksu przyznanego wartości `b`,
- jeżeli `funkcja(a, b)` zwróci wartość równą `0`, to wartości indeksów pozostaną bez zmian,
- jeżeli `funkcja(a, b)` zwróci wartość większą od `0`, to wartości `a` zostanie nadany indeks większy od indeksu przyznanego wartości `b`.

Przykład 3.66

```
function porownaj(a, b) {
  return a - b;
}
var Tablica = new Array(27, 100, 10, 450, 1654, 320);
document.write('Bez sortowania: ' + Tablica.join());
document.write("<br>" + 'Sortowanie domyślne: ');
Tablica.sort();
```

```
document.write(Tablica.join());
document.write("<br>" + 'Sortowanie poprawne: ');
Tablica.sort(porownaj);
document.write(Tablica.join());
```

W podanym przykładzie została zdefiniowana funkcja `porownaj(a, b)`, która zwróci wartość mniejszą od zera, równą zero lub większą od zera. W zależności od zwróconej wartości elementy tablicy zostaną poprawnie uporządkowane od wartości najmniejszej do największej.

Zadanie 3.2

Utwórz tablicę, której elementy będą zawierały cyfry i litery. Zdefiniuj funkcję, która pozwoli uporządkować elementy tablicy w ten sposób, że będą w niej umieszczone najpierw litery w kolejności alfabetycznej, a następnie cyfry w kolejności od wartości najmniejszej do największej.