Lekcja 2

Temat: Wartości, zmienne, stałe. Typy Danych

Instrukcje JavaScript składają się z:

wartości, operatorów, wyrażeń, słów kluczowych i komentarzy.

- Większość programów JavaScript zawiera wiele instrukcji JavaScript.
- Oświadczenia są wykonywane jeden po drugim, w tej samej kolejności, w jakiej zostały napisane.
- Średniki oddzielają instrukcje JavaScript.
- Dodajemy średnik na końcu każdej instrukcji wykonywalnej.
- JavaScript rozróżnia wielkość liter
- W przypadku rozdzielenia średnikami dozwolonych jest wiele instrukcji w jednym wierszu:

```
a = 5; b = 6; c = a + b;
```

Uwaga : W JavaScript wszystkie instrukcje kodu powinny kończyć się średnikiem (;) — Twój kod może działać poprawnie dla pojedynczych linii, ale prawdopodobnie nie będzie działał, gdy piszesz wiele linii kodu razem. Postaraj się wyrobić w sobie nawyk włączania go.

Zmienne i stałe JavaScript

W języku JavaScript uznaje się, że każdy dostarczany lub używany fragment zawiera wartość. Na przykład w instrukcji:

```
alert("Witaj, świecie!");
```

słowa *Witaj, świecie!* stanowią wartość przekazywanego komunikatu. W świecie JavaScriptu, każdy wykorzystywany kawałek danych jest uważany za wartość.

Podczas pracy z wartościami potrzebujemy dwóch rzeczy, aby ułatwić sobie życie. Musimy:

1. Łatwo je identyfikować.
2. Wykorzystywać je ponownie w całej aplikacji bez niepotrzebnego duplikowania samej wartości.

Te dwie rzeczy są zapewniane przez **zmienne.**
W programowaniu zmienna jest pojemnikiem (obszarem przechowywania) do przechowywania danych.

Zmienna jest po prostu nazwą dla wartości. Zamiast wpisywać *Witaj, świecie!* za każdym razem, gdy chcemy użyć tego wyrażenia w aplikacji, możemy przypisać je do zmiennej i używać tej zmiennej zawsze, gdy ponownie potrzebujemy użyć *Witaj, świecie!*.

Aby użyć zmiennej, musisz ją najpierw utworzyć — dokładniej nazywamy to zadeklarowaniem zmiennej.

Zmienne - Starsze rozwiązanie, użycie słowa kluczowego var:

Zmiennych używamy poprzez zastosowanie słowa kluczowego **`var`**, po którym następuje nazwa, jaką chcemy nadać danej zmiennej:

```
var myText;
```

W tej linii kodu zadeklarowaliśmy zmienną o nazwie `myText`. Na tę chwilę nasza zmienna została po prostu zadeklarowana. Nie zawiera żadnej wartości, stanowi pusty pojemnik. Po wpisaniu nazwy zmiennej powinieneś otrzymać zwróconą wartość undefined. (Jeśli nie istnieją, otrzymasz komunikat o błędzie)
Naprawmy to, inicjując zmienną z wartością, taką jak na przykład *Witaj, świecie!*:

```
var myText = "Witaj, świecie!";
```

Dodanie instrukcji alert spowoduje wyświetlenie tekstu na stronie:

```
var myText = "Witaj, świecie!";
alert(myText);
```

Zwróć uwagę, że nie przekazujemy już tekstu *Witaj, świecie!* do funkcji alert w sposób bezpośredni. Zamiast tego przekazujemy zmienną o nazwie myText.
Efekt tej zmiany jest prosty. Pozwala to określić wyrażenie *Witaj, świecie!* tylko w jednym miejscu w kodzie. Jeśli chcielibyśmy zmienić ten tekst na *Pies zjadł moją pracę domową!*, musielibyśmy po prostu dokonać jedynie zmiany wartości określonej przez zmienną myText:

```
var myText = "Pies zjadł moją pracę domową!";
alert(myText);
```

Gdziekolwiek w kodzie odwołasz się teraz do zmiennej myText, użyta zostanie ta nowa wartość.

Przykład 1. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l2p1.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Zmienne - l2p1 </title>
    </head>
    <body>
        <h2> Zmienne</h2>
        <script type="text/javascript">
            var myText = "Witaj, świecie!";
            alert(myText);
            document.write(myText);
            document.write("I cóż, ponownie:");
            document.write(myText);
        </script>
    </body>
</html>
```

Zmień tekst przypisany do zmiennej i ponownie uruchom stronę.

Inne sposoby deklaracji zmiennych

Nie zawsze trzeba używać słowa kluczowego var, aby zadeklarować zmienną. Można po prostu zrobić to następująco:

```
myText = "Witaj, świecie!";
alert(myText);
```

Zmienna `myText` została użyta tutaj bez formalnego zadeklarowania jej za pomocą słowa kluczowego `var`. Chociaż nie jest to zalecane, to jest całkowicie w porządku. Jednak poprzez zadeklarowanie zmiennej w ten sposób deklarujemy ją **globalnie**. Przyjrzymy się znaczeniu określenia „globalnie" podczas późniejszych lekcji.

Należy wspomnieć o jeszcze jednej kwestii. Należy pamiętać o tym, że deklaracja i inicjowanie zmiennej nie muszą być częścią tego samego polecenia. Można podzielić je na wiele poleceń:

```
var myText;
myText = "Witaj, świecie!";
alert(myText);
```

W praktyce takie rozdzielenie deklaracji i inicjowania zmiennych jest bardzo częste.

Przykład 2. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l2p2.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Sposoby deklaracji zmiennych - l2p2 </title>
    </head>
    <body>
        <script type="text/javascript">
            var zmiennal; //deklaracja zmniennej, wartość przypisujemy później
            zmiennal=5; // przypisanie wartości

            var zmienna2=10; //deklaracja wraz z  przypisaniem wartości

            zmienna3=30; // bez var
            alert(zmiennal); //alerty
            alert(zmienna2);
            alert(zmienna3);
        </script>
        <h2> Zmienne</h2>
    </body>
</html>
```

Możesz zmieniać wartość zmiennej, kiedy tylko chcesz, i nadawać jej inną, całkowicie dowolną wartość:

```
var myText;
myText = "Witaj, świecie!";
myText = 99;
myText = 4 * 10;
myText = true;
myText = undefined;
alert(myText);
```

W powyższych przykładach użyto różnych wartości: tekst, liczby, wartość logiczna itp. Możliwe jest to dlatego, że **JS umożliwia przechowywanie w zmiennych różnych typów danych.**

Przykład 3. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l2p3.html.*

Wprowadź kod bezpośrednio w konsoli:

```
var txt = "Ala ma kota";
var nr = 20;
var url = "kontakt.html";

console.log(txt);
console.log(nr);
console.log(url);
```

lub użyj jako stronę zawierającą skrypt:

```html
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Sposoby deklaracji zmiennych - l2p3 </title>
    </head>
    <body>
        <h2> Zmienne</h2>
        <script>
            var txt = "Ala ma kota";
            var nr = 20;
            var url = "kontakt.html";

            console.log(txt);
            console.log(nr);
            console.log(url);
        </script>
    </body>
</html>
```

```
var orderValue = 100;
orderValue = 110;
orderValue = 210;
var orderValue = 320;
console.log(orderValue);
```

Zakres zmiennej ograniczony do wnętrza funkcji

```
function showOrderValue() {
    var orderValue = 100;
    console.log(orderValue);
}

showOrderValue();
console.log(orderValue);
```

Zmienna globalna, dostępna wszędzie:

```
var orderValue = 100;

function showOrderValue() {
    console.log(orderValue);
}

showOrderValue();
console.log(orderValue);
```

Deklaracja zmiennych i stałych w ES6 - nowsze rozwiązania:

Wraz z pojawieniem się ES6 (ECMAScript 2015) wprowadzono nowe sposoby deklaracji zmiennych i stałych za pomocą słów kluczowych **`let`** i **`const`**.

Słowo **`let`** oznacza zmienną, natomiast **`const`** stałą (taką, do której po ustawieniu wartości, nie możemy przypisać nowej).

Przykład 4. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l2p4.html.*

Wprowadź kod bezpośrednio w konsoli:

```
let txt = "Przykładowy tekst"; //zmienna
txt = "Inny tekst"; //zmieniamy wartość
console.log(txt);

const nr = 102; //stała
nr = 103; //błąd - nie można zmieniać stałej
console.log(nr);
```

lub użyj jako stronę zawierającą skrypt:

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Sposoby deklaracji zmiennych - l2p4 </title>
    </head>
    <body>
        <h2> Zmienne</h2>
        <script>
            let txt = "Przykładowy tekst"; //zmienna
            txt = "Inny tekst"; //zmieniamy wartość
            console.log(txt);

            const nr = 102; //stała
            nr = 103; //błąd - nie można zmieniać stałej
            console.log(nr);
        </script>
    </body>
</html>
```

Różnica między var i let

Kiedy po raz pierwszy stworzono JavaScript, istniał tylko var. W większości przypadków działa to dobrze, ale ma pewne problemy ze sposobem działania — skrypt może czasami być mylący lub wręcz denerwujący. Tak więc let został stworzony we współczesnych wersjach JavaScript, nowe słowo kluczowe do tworzenia zmiennych, które działa nieco inaczej niż var, naprawiając pewne problemy.

1. Jeśli piszesz wielowierszowy program JavaScript, który deklaruje i inicjuje zmienną, możesz faktycznie zadeklarować zmienną za pomocą var po jej zainicjowaniu i nadal będzie działać. Na przykład:

```
myName = 'Chris';
function logName() {
  console.log(myName);
}

logName();
var myName;
```

Uwaga : to nie zadziała podczas wpisywania pojedynczych wierszy do konsoli JavaScript, tylko podczas uruchamiania wielu wierszy JavaScript w dokumencie internetowym.

Działa to dzięki **podnoszeniu (**hosting)

Hosting – wciąganie, deklaracje zmiennej wciągane są na początek bloku, ich wartości zostają przypisane później, zgodnie z kolejnością.

Hoisting nie działa już z let. Gdybyśmy zmienili var na let w powyższym przykładzie, zakończyłoby się niepowodzeniem z błędem. To dobrze — zadeklarowanie zmiennej po jej zainicjowaniu skutkuje mylącym, trudniejszym do zrozumienia kodem.

var:

```
function showVariable() {
    var x = 2;
    console.log(x);
}

showVariable();
```

```
function showVariable() {
    console.log(x);
    var x = 2;
}

showVariable();
```

```
undefined
>
```

Hoisting w JS działa także w przypadku funkcji, można najpierw wywołać funkcję a dopiero później ją zdefiniować (nie działa w bardziej restrykcyjnych językach)

```
1  showVariable();
2
3  function showVariable() {
4      console.log(x);
5      var x = 2;
6  }
```

let i const:

```
1  showVariable();
2
3  function showVariable() {
4      console.log(x);
5      let x = 2;
6  }
```

```
1  showVariable();
2
3  function showVariable() {
4      console.log(x);
5      const x = 2;
6  }
```

```
▶Uncaught ReferenceError: x is not defined
    at showVariable (script.js:4)
    at script.js:1
```

Nie występuje hoinsting, dlatego używając let i const najpierw deklarujemy zmienne lub stałe i dopiero później z nich korzystamy.

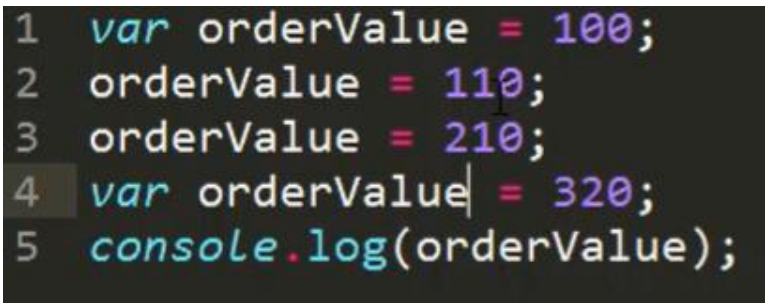
```
1  showVariable();
2
3  function showVariable() {
4      const x = 2;
5      console.log(x);
6  }
```

2. używając var, możesz zadeklarować tę samą zmienną tyle razy, ile chcesz, w let nie możesz. Działałoby to:

```
var myName = 'Chris';

var myName = 'Bob';
```

i to:

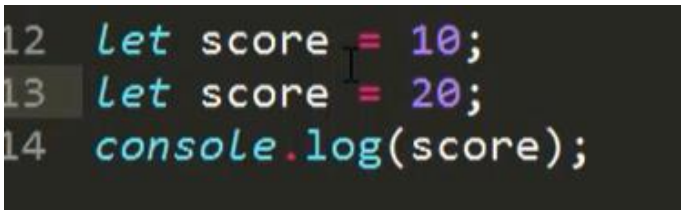


Ale poniższy kod spowoduje błąd w drugiej linii:

```
let myName = 'Chris';

let myName = 'Bob';
```

Ten kod też nie zadziała:



Poprawnie zadziała kod:

```
let myName = 'Chris';

myName = 'Bob';
```

Zatem let zabezpiecza przed ponownym zadeklarowaniem tej samej zmiennej w danym bloku

Z tych i innych powodów zalecamy używanie let w kodzie jak największej ilości, a nie var.

Nie ma powodu, aby używać var, chyba że musisz obsługiwać stare wersje Internet Explorera za pomocą swojego kodu (nie obsługuje on let do wersji 11; przeglądarka Microsoft Edge obsługuje let dobrze).

3. Let – zmienne zadeklarowane mają ograniczenie tylko do bloków wewnątrz których powstały. Pozwala to ściśle kontrolować zakres zmiennej, nie ma ryzyka nadpisania wartości zmiennej poza blokiem

```
1
2  function showOrderValue() {
3      var orderValue = 100;
4      if (orderValue >= 100) {
5          let newOrderValue = orderValue - orderValue * 0.1;
6      }
7      console.log(newOrderValue);
8  }
9
10 showOrderValue();
11
```

Można zadeklarować zmienną o tej samej nazwie w różnych blokach

```
let score = 10;
let x = 1;
if (x == 1) {
    let score = 20;
    console.log(score);
}
```

Stałe:

const - Deklarujemy stałe, przypisujemy do niej raz wartość, której nie możemy zmienić

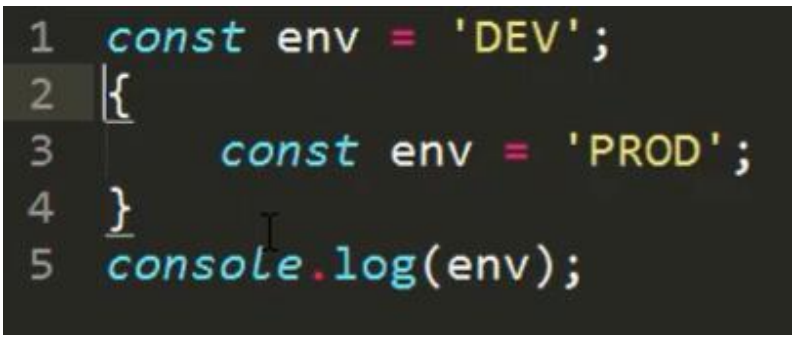
```
1  const env = 'DEV';
2  env = 'PROD';
3  console.log(env);
```

```
▶Uncaught TypeError: Assignment to constant variab
    at script.js:2
```

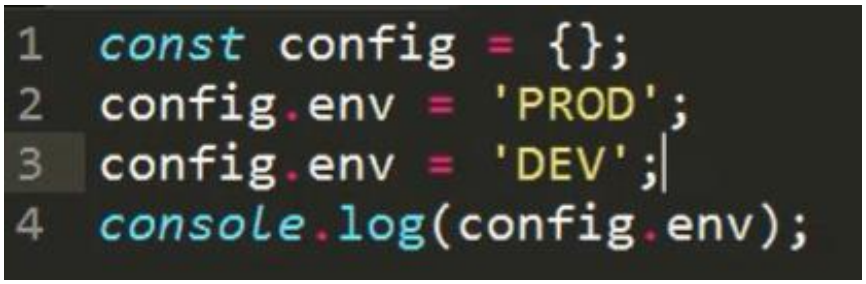
Podobnie jak zmiennej let, stałej const nie można ponownie zadeklarować

```
1  const env = 'DEV';
2  const env = 'PROD';
3  console.log(env);
```

Możemy dokonać ponownej deklaracji w innym bloku.



Gdy, do stałej przypisany jest obiekt, nie możemy zmienić obiektu, ale bez problemu można dokonać zmian właściwości obiektu



Nazywanie zmiennych

Mamy dużo swobody w nazywaniu zmiennych według własnych upodobań. Język JavaScript jest bardzo wyrozumiały w kwestii znaków (liter, cyfr i innych symboli generowanych za pomocą klawiatury), jakie mogą znaleźć się w nazwie zmiennej.
Podczas nazywania zmiennych należy zasadniczo pamiętać o następujących rzeczach:

1. Nazwy zmiennych mogą być dowolnie krótkie (na przykład zawierać jeden znak) lub dowolnie długie.
2. Nazwy zmiennych mogą zaczynać się od litery, podkreślenia (_) lub znaku dolara ($).
   Nie mogą zaczynać się od cyfry.
3. Poza pierwszym znakiem, który nie może być cyfrą, pozostałe znaki nazw zmiennych mogą tworzyć dowolną kombinację liter, podkreśleń, cyfr i znaków $. Możesz również do woli łączyć ze sobą i mieszać małe i wielkie litery.
4. Spacje są niedozwolone.
5. Nie można używać również nazw, które są słowami kluczowymi JavaScript (np. alert, function itp.)
6. **Nazywaj swoje zmienne tak, by dało się zrozumieć do czego się odnoszą**.
7. Bezpieczną konwencją, której należy się trzymać, jest tak zwana camelCase , gdzie sklejasz wiele słów, używając małych liter dla całego pierwszego słowa, a następnie pisząc wielkie litery.
8. Dobrą praktyką jest używanie angielskich nazw dla zmiennych

Poniżej kilka przykładów poprawnych nazw zmiennych:

```
var myText;
var $;
var r8;
var _counter;
var $field;
var thisIsALongVariableName_butItCouldBeLonger;
var __$abc;
```

Typy Danych

Podstawowymi typami w języku JavaScript są: **String (łańcuchy znaków), Number (liczby), Boolean (wartości logiczne), Null (typ pusty), Undefined (typ nieokreślony) i Object (obiekty)**.

Tabela Podstawowe typy JavaScript

Typ	**Co robi**
String	Podstawowa struktura do pracy z tekstem
Number	Jak można zgadnąć, pozwala pracować z liczbami
Boolean	Powstaje, gdy używamy true i false
Null	Reprezentuje cyfrowy odpowiednik niczego...
Undefined	Choć trochę podobny do null, jest zwracany, gdy wartość powinna istnieć, ale nie istnieje... tak jak w sytuacji, w której deklarujemy zmienną, ale niczego do niej nie przypisujemy
Object	Działa jak opakowanie dla innych typów, w tym innych obiektów

Łańcuch znaków (String)

Z **łańcuchami znaków** można pracować, używając ich po prostu w kodzie. Należy się tylko upewnić, że umieściło się je w pojedynczym lub podwójnym cudzysłowie. Poniżej znajduje się kilka przykładów:

```
var text = "to jest jakiś tekst";
var moreText = 'jestem umieszczony w pojedynczym
cudzysłowie!';
alert("to jakiś kolejny tekst");
```

Łączenie (czyli konkatenacja) łańcuchów znaków

Poza zwykłym wyliczaniem łańcuchów znaków często będziemy **łączyć ze sobą kilka z nich**. Można to łatwo zrobić za **pomocą operatora konkatenacji: +** lub **+=**

```
var initial = "Witaj,";
alert(initial + " świecie!");
alert("Mogę również " + "zrobić tak!");


let stringA = "Jestem prostym łańcuchem znaków.";
let stringB = "Ja też jestem prostym łańcuchem znaków!";
alert(stringA + " " + stringB);
```

Zwróć uwagę, że w trzeciej linii dodajemy do siebie stringA i stringB. Pomiędzy nimi określamy pusty znak spacji (" "), aby upewnić się, że pomiędzy tymi łańcuchami znaków będzie odstęp.

Łańcuchy szablonowe- alternatywa dla konkatenacji łańcuchów z wykorzystaniem +

Tradycyjnie:

```
let price = 199;
let currency = 'PLN';
let label = 'Cena wynosi: ' + price + currency;
console.log(label);
```

```
Cena wynosi: 199PLN
```

Nowe rozwiązanie:

```
let price = 199;
let currency = 'PLN';
let label = `Cena wynosi: ${price}${currency}`
console.log(label);
```

Łańcuchy szablonowe pozwalają na dynamiczne wstawianie wyrażeń:

```
let price = 199;
let currency = 'PLN';
let discount = 0.1;
let label = `Cena wynosi: ${price - price * discount}${currency}`
console.log(label);
```

Typ liczbowy

Typ liczbowy służy do reprezentacji liczb. Liczby zapisywane są za pomocą ciągów znaków składających się na liczbę, np. 24 (umieszczony w kodzie skryptu tekst 24 to dwa znaki, dwójka i czwórka, które razem stanowią literał — stałą napisową — interpretowany jako liczba 24). Obowiązują przy tym następujące zasady:

- Jeżeli ciąg cyfr nie jest poprzedzony żadnym znakiem lub jest poprzedzony znakiem *+*, reprezentuje wartość dodatnią, jeżeli natomiast jest poprzedzony znakiem –, reprezentuje wartość ujemną.
- Jeżeli literał rozpoczyna się od cyfry *0*, jest traktowany jako wartość ósemkowa.
- Jeżeli literał rozpoczyna się od ciągu znaków 0x, jest traktowany jako wartość szesnastkowa (heksadecymalna). W zapisie wartości szesnastkowych mogą być wykorzystywane zarówno małe, jak i wielkie litery alfabetu, od *A* do *F*.
- Literały mogą być zapisywane w notacji wykładniczej, w postaci X.YeZ, gdzie X to część całkowita, Y — część dziesiętna, natomiast Z to wykładnik potęgi liczby 10. Zapis taki oznacza to samo, co X.Y * 10Z. **W liczbach rzeczywistych separatorem dziesiętnym jest kropka** (zamiast występującego w notacji polskiej przecinka, np. zapis 3.14 oznacza wartość 3,14, czyli trzy i czternaście setnych).

Przykłady literałów liczbowych:

123— dodatnia całkowita wartość dziesiętna 123,
-123— ujemna całkowita wartość dziesiętna –123,
012— dodatnia całkowita wartość ósemkowa równa 10 w systemie dziesiętnym,
-024— ujemna całkowita wartość ósemkowa równa 20 w systemie dziesiętnym,
0xFF— dodatnia całkowita wartość szesnastkowa równa 255 w systemie dziesiętnym,
-0x0f— ujemna całkowita wartość szesnastkowa równa –15 w systemie dziesiętnym,
1.1— dodatnia wartość rzeczywista 1,1,
-1.1— ujemna wartość rzeczywista –1,1,
0.1E2— dodatnia wartość rzeczywista równa 10,
1.0E-2— dodatnia wartość rzeczywista równa 0,01.

Przykład 3. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l2p3.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Zmienne, typy, łańcuchy znaków - l2p3 </title>
    </head>
    <body>
        <script>
            //konkatenacja, różne typy
            var initial = "Witaj,";
            alert(initial + " świecie!");
            alert("Mogę również " + "zrobić tak!");
            var myText;
            myText = 40;
            alert(myText);

            var stringA = "Jestem prostym łańcuchem znaków.";
            var stringB = "Ja też jestem prostym łańcuchem znaków!";
            alert(stringA + " " + stringB);

            // polskie nazwy zmiennych
            var lata="Mam lat "+25;
            document.write(lata);
        </script>
    </body>
</html>
```

Sekwencje znaków specjalnych

Uwaga:

Jeżeli napis ma zawierać znak specjalny, to trzeba poprzedzić go lewym ukośnikiem (ang. backslash)

Znaki cudzysłowu (apostrofu) są tu wyróżnikami ciągu (napisu). Należy na to zwrócić uwagę, gdy sam ciąg ma zawierać jeden z tych znaków. Jeżeli w ciągu ma wystąpić znak użyty jako wyróżnik ciągu, znak ten musi być zastąpiony przez sekwencję specjalną.

Napisy mogą zawierać **sekwencje znaków specjalnych** przedstawione w tabeli:

Sekwencja	Znaczenie
\b	backspace (ang. backspace)
\n	nowy wiersz (ang. new line)
\r	powrót karetki (ang. carriage return)
\f	nowa strona (ang. form feed)
\t	tabulacja pozioma (ang. horizontal tab)

\"	cudzysłów (ang. double quote)
\'	apostrof (ang. single quote)
\\	lewy ukośnik (ang. backslash)

Przykład 4. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l2p4.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Sekwencje specjalne - l2p4 </title>
    </head>
    <body>
        <script>
            // Wyróżnikami ciągu są znaki cudzysłowu.
            // Konieczne jest użycie sekwencji specjalnej \"
            document.write("<p>To jest cytat: \"Być albo nie być...\".</p>");

            // Wyróżnikami ciągu są znaki apostrofu.
            // Konieczne jest użycie sekwencji specjalnej \'
            document.write('<p>Prawa Murphy\'ego to zbiór humorystycznych powiedzeń.</p>');

            // Sekwencji specjalnych nie trzeba używać po zamianie
            // znaków wyróżniających ciąg.
            document.write('<p>To jest cytat: "Być albo nie być...".<p>');
            document.write("<p>Prawa Murphy'ego to zbiór humorystycznych powiedzeń.<p>");
        </script>
    </body>
</html>
```

Ćwiczenie nr 1

Jaki będzie efekt działania skryptu?

```
<body>
    <h3>
        Generowanie treści na stronie
    </h3>

    <p id="output1"></p>
    <p id="output2"></p>
    <p id="output3"></p>
    <p id="output4"></p>
    <p id="output5"></p>
    <script>
    let number1=5;
    let number2=4;
    let text1="4";
    document.getElementById("output1").innerHTML = "5+4= " + number1 + number2;
    //document.getElementById("output2").innerHTML = number1 + number2;
    //document.getElementById("output3").innerHTML ="5+4= "+ number1 + text1;
    //document.getElementById("output4").innerHTML =number1 + text1;
    //document.getElementById("output5").innerHTML =`5+4=  ${number1 +
    number2}`;
    </script>
</body>
```