**String** jest typem danych służącym do przechowywania ciągu znaków. Ciąg znaków zapisujemy w cudzysłowia i przypisujemy do zmiennej za pomocą znaku przypisania (=).

Ciąg znaków działa jak tablica przechowująca poszczególne znaki w kolejnych swoich komórkach. Zatem jak w przypadku zwykłych tablic, możemy użyć na ciągu znaków własności **length**, aby sprawdzić długość ciągu znaków.

Przykład definicji ciągu znaków oraz wypisania jego długości na ekran:

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write( tekst.length ); // 43

Ciągi znaków można łączyć ze sobą, w jeden łańcuch. Jest to tzw. konkatenacja. Dokonujemy tego za pomocą symbolu +.

Przykład konkatenacji. Łączymy ze sobą ciąg znaków zawarty w zmiennej tekst oraz dodatkowy łańcuch zawarty w apostrofach. Wynik naszego działania wypisujemy w konsoli przeglądarki.

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

tekst = tekst + ' Nieprawdaż?';

document.write(tekst); // Wszędzie dobrze, ale w 127.0.0.1 najlepiej. Nieprawdaż?

Aby wielokrotnie dopisać jakieś powtarzalne elementy do naszego łańcucha znaków możemy użyć mechanizmu pętli. W poniższym przykładzie trzykrotnie dopiszemy do początkowo pustej zmiennej **zdanie** zawartość zmiennej **slowo**.

const slowo = "Localhost ";

var zdanie = "";

for (let i=0; i<3; i++) {

zdanie += slowo;

}

document.write(zdanie); // Localhost Localhost Localhost

Pisałem wcześniej, że łańcuch znaków, to tak naprawdę tablica zawierająca osobno poszczególne znaki. Możemy się do nich odwołać wykorzystując metodę **charAt**(indeks), gdzie indeks jest numerem danego znaku w ciągu znaków, bądź w sposób typowo tablicowy – podając wartość indeksu wewnątrz nawiasów kwadratowych. Pamiętajmy, że numeracja rozpoczyna się od 0. Zatem poniższym przykładzie literka W rozpoczynająca zdanie ze zmiennej tekst będzie miała indeks 0, kolejna litera (s) indeks 1, i tak dalej.

Odwołując się do konkretnego indeksu, możemy używać działań matematycznych, np. odwołać się do indeksu o numerze 1+2 (3), bądź do indeksu tekst.length – x, gdzie x będzie numerem znaku licząc od końca ciągu.

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write(tekst.charAt(0)); // W

document.write(tekst.charAt(6)); // i

document.write(tekst[0]); // W

document.write(tekst[6]); // i

document.write(tekst.charAt(tekst.length-3)); // e

document.write(tekst[tekst.length-3]); // e

Każdy znak posiada swój unikatowy numer, kod w siedmiobitowym systemie ASCII. Aby odczytać kod ASCII danego znaku, wykorzystamy metodę **charCodeAt**(). Dowiemy się, że litera W ma kod87, natomiast i 105. Odwrotną pracę, czyli zamianę kodu na znak dokona metoda lasy String, String.**fromCharCode**(x), gdzie x to kod znaku.

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write(tekst.charCodeAt(0)); // 87

document.write(tekst.charCodeAt(6)); // 105

document.write("Wartość kodu 110: " + String.fromCharCode(110)); // n

Wykorzystując dwie kolejne metody, możemy zamienić litery w ciągu znaków na wielkie lub małe. I tak odpowiednio: **toUpperCase** zamieni nam wszystkie litery w ciągu na wielkie, natomiast **toLowerCase**, na małe.

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write(tekst.toUpperCase()); // WSZĘDZIE DOBRZE, ALE W 127.0.0.1 NAJLEPIEJ

document.write(tekst.toLowerCase()); // wszędzie dobrze, ale w 127.0.0.1 najlepiej.

Możemy wykorzystać znajomość powyższych metod i zamienić w zdaniu jedynie pierwszą literę na wielką. W tym celu wybieramy pierwszą literę (charAt(0)), zamieniamy ją na wielką (toUpperCase), a następnie dopisujemy resztę zawartości zmiennej, pozbawioną pierwszej litery. Wykorzystujemy do tego metodę .slice z parametrem 1, pozostawieniem ciągu znaków od znaku z indeksem 1 (bez 0, pierwszej litery).

var tekst = "wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write( tekst.charAt(0).toUpperCase() + tekst.slice(1) ); // Wszędzie dobrze, ale w 127.0.0.1 najlepiej.

Sprawdzimy tera, na jakiej pozycji w tekście znajduje się poszukiwany przez nas ciąg znaków. Wykorzystamy do tego metodę **indexOf**(“”) zwracającą indeks pierwszej litery poszukiwanego ciągu, jeśli takowy został odnaleziony, bądź -1 jeśli go nie ma.

Metoda **lastIndexOf**(“”) zwraca natomiast indeks pierwszej litery ostatniego znalezionego szukanego ciągu (jeśli występuje wielokrotnie).

var tekst = "wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write(tekst.indexOf("ale")); // 17

document.write(tekst.lastIndexOf("e")); // 40

Czas wyciąć określony fragment tekstu. Wykorzystamy do tego metodę **substr**(start, długość), gdzie start jest indeksem litery, od której rozpoczniemy wycinany fragment, natomiast długość mówi nam ile znaków wycinamy.

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write(tekst.substr(0)); // Wszędzie dobrze, ale w 127.0.0.1 najlepiej.

document.write(tekst.substr(0, 5)); // Wszęd

document.write(tekst.substr(8, 3)); // do

document.write(tekst.substr(6, tekst.length - 7)); // ie dobrze, ale w 127.0.0.1 najlepiej

Inną opcją na wycięcie fragmentu ciągu znaków jest metoda **substring**(początek, koniec), gdzie podajemy indeksy znaku początkowego i końcowego wycinanego tekstu. Jeśli wartości początku i końca wpiszemy odwrotnie, to zostaną one automatycznie zamienione miejscami. Natomiast jeśli nie podamy wartości końcowej, zostanie pobrany tekst od podanego indeksu znaku, aż do końca.

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write(tekst.substring(0, 3)); // Wsz

document.write(tekst.substring(3)); // ędzie dobrze, ale w 127.0.0.1 najlepiej.

document.write(tekst.substring(7, 4)); // dzi

Kolejną możliwością wycinania tekstu jest metoda **slice**(początek, koniec), której działanie jest prawie identyczne, jak substring(). Jednak w tym przypadku, jeśli wartości start i stop podamy odwrotnie, nie zostaną one automatycznie zamienione i uzyskamy pusty ciąg znaków. Natomiast jeśli podamy wartość ujemną parametru start lub koniec, będziemy go liczyć od końca ciągu znaków.

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

const tekst2 = tekst.slice(0,4);

document.write(tekst2); // Wszę

const tekst3 = tekst.slice(1,6);

document.write(tekst3); // szędz

const tekst4 = tekst.slice(3,7);

document.write(tekst4); // ędzi

const tekst5 = tekst.slice(9);

document.write(tekst5); // dobrze, ale w 127.0.0.1 najlepiej.

const tekst6 = tekst.slice(-3);

document.write(tekst6); // ej.

Tekst możemy również podzielić na elementy. Będą one rozdzielone na fragmenty oddzielone przez określony znak lub ciąg znaków. I tak na przykład możemy podzielić akapit na zdania, czy jak w poniższym przykładzie, zdanie złożone na poszczególne fragmenty, za pomocą symbolu przecinka. Wykorzystamy również mechanizm pętli, o którym wkrótce opowiemy sobie więcej. Wykorzystamy metodę **split**(“element\_dzielący”).

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

const elementy = tekst.split(", ");

for (let i=0; i<elementy.length; i++) {

document.write(elementy[i]);

}

/\*

Wszędzie dobrze

ale w 127.0.0.1 najlepiej.

\*/

Metoda **replace**(“co”, “na co”) zamienia nam zadany fragment tekstu na inny. Przykładowo zamieńmy sobie adres hosta lokalnego na słówko “domu”:

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

const zmienione = tekst.replace("127.0.0.1", "domu");

document.write(zmienione); // Wszędzie dobrze, ale w domu najlepiej.

Ostatnimi z omawianych przeze mnie dziś metod będą **encodeURI** oraz **decodeURI**, służące do zamiany tekstu na adres URI i odwrotnie. Zamienione zostają specyficzne litery językowe, jak np. polskie znaki, spacje oraz znaki specjalne. Przykład poniżej:

var tekst = "Wszędzie dobrze, ale w 127.0.0.1 najlepiej.";

document.write(encodeURI(tekst)); // Wsz%C4%99dzie%20dobrze,%20ale%20w%20127.0.0.1%20najlepiej.

tekst = encodeURI(tekst);

document.write(decodeURI(tekst)); // Wszędzie dobrze, ale w 127.0.0.1 najlepiej.