Programowanie w JavaScript

WSEI

#szkoła programowania

W JavaScript tak samo jak w innych językach możemy operować nie tylko na liczbach, ale także na tekstach. Tak zwane stringi w JavaScript możemy tworzyć na kilka sposobów:

```
const text = "Ala ma kota, a kot ma Ale.";

const text = 'Ala ma kota';

const text = `Ala ma kota`;

const img = '<div class="element" data-text="test">';

const txt = "It's a big year";
```

Stringi na wiele linii

Tworząc wieloliniowe teksty możemy zastosować kilka technik.

```
//poprzez operator przypisania
let text = "Stoi na stacji lokomotywa,<br>";
text += "Ciężka, ogromna i pot z niej spływa:<br>";
text += "Tłusta oliwa.";

//poprzez dodawanie części tekstu
let text = "Stoi na stacji lokomotywa,<br>"
+ "Ciężka, ogromna i pot z niej spływa:<br>"
+ "Tłusta oliwa.";

//Poprzez zastosowanie znaku backslash na końcu linii
let text = "Stoi na stacji lokomotywa,<br>\
Ciężka, ogromna i pot z niej spływa:<br>\
Tłusta oliwa.\
";

//Najlepsza metoda - użycie template strings
let text = `Stoi na stacji lokomotywa,<br>
Ciężka, ogromna i pot z niej spływa:<br>
Tłusta oliwa.
`;
```

Stringi – wstawianie zmiennych

Do wnętrza tekstu zmienne możemy wstawić na kilka sposobów. Dwa najpopularniejsze to dodawanie do stringa zmiennej oraz w przypadku template strings użycie tak zwanej interpolacji:

```
const age = 10;

const text = "Ten pies ma " + age + " lat";
const text = 'Ten kot ma ' + age + ' lat';
const text = `Ten chomik ma ${age} lat`;

const a = 112;
const b = 120;

const text = "Cena produktu A to " + a + "zł, cena produktu B to " + b + "zł, a suma to " + (a+b)+ "zł";
const text = `Cena produktu A to ${a}zł, cena produktu B to ${b}zł, a suma to ${a+b}zł`;
```

Aby odczytać długość naszego tekstu posłużymy się właściwością **length**.

Do pobrania w tekście znaku na danej pozycji możemy zastosować dwa podejścia. Jedno z nich to użycie metody charAt(). Drugie - to odwoływanie się do liter tekstu jak do elementów tablicy - poprzez kwadratowe nawiasy.

```
const text = "Ala";
text.length; //3

const text = "Ala ma kota, a kot ma Ale";

console.log(text.charAt(0)); //A
console.log(text.charAt(4)); //m

console.log(text[0]); //A
console.log(text[4]); //m

console.log(text.charAt(text.length-1)); //e
console.log(text[text.length-1]); //e
```

Stringi – duże i małe literki

Metody **toUpperCase()** i **toLowerCase()** służą odpowiednio do zamieniania tekstu na duże i małe litery.

```
const text = "Ala ma kota";

console.log(text.toUpperCase()); //"ALA MA KOTA"
console.log(text.toLowerCase()); //"ala ma kota"
```

Stringi – szukanie pozycji fragmentu

Metoda **indexOf** służy do podawania pozycji szukanego fragmentu w tekście (ale także w tablicy, bo metoda ta dostępna jest dla stringów i tablic). Jeżeli zwróconą wartością jest -1, to szukanego tekstu nie ma:

```
"Ala ma kota".indexOf("kot"); //7
```

Podobne działanie ma **metoda lastIndexOf**, podaje ona jednak numer ostatniego wystąpienia podtekstu

```
"Ala ma kota i tak już jest".lastIndexOf("a"); //15 - bo ostatnia litera występuje na pozycji 15
```

Stringi – jak się zaczyna i jak się kończy

```
const text = "Ala ma kota";

text.startWith("Ala"); //true
text.startWith("Ola"); //false

text.endsWith("kota"); //true
text.endsWith("psa"); //false
```

Metoda **substr(start, lng)** służy do zwracania kawałka tekstu. Pierwszym jej parametrem jest początek pobieranego kawałka tekstu, a drugi opcjonalny wskazuje długość pobieranego tekstu. Jeżeli drugi parametr nie zostanie podany, wówczas pobierany kawałek będzie pobierany do końca tekstu.

```
const text = "Ala ma kota";

console.log(text.substr(2)); //"a ma kota"
console.log(text.substr(0, 3)); //"Ala"
console.log(text.substr(7, 4)); //"kota"
console.log(text.substr(4, text.length - 4)); //wypisze tekst od 4 litery do końca - "ma kota"
```

Metoda **substring(start, stop)** ma bardzo podobne działanie co powyższa. Różnicą jest drugi parametr, który zamiast długości wyznacza miejsce końca pobieranego kawałka.

```
console.log(text.substring(0, 3)); //"Ala"
console.log(text.substring(3)); //"ma kota"
console.log("Ala ma kota".substring(6, 4)); //"ma"
```

Stringi – slice

Tak samo jak w przypadku tablic, tak i w przypadku zmiennych tekstowych możemy skorzystać z metody **slice(start, stop),** która zwraca nam kawałek tekstu. Jej działanie jest praktycznie identyczne do działania metody **substring()**, jednak występują małe różnice. Jeżeli drugi argument będzie mniejszy od pierwszego, wtedy w przeciwieństwie do **substring()** argumenty nie zostaną zamienione miejscami.

```
const txt = "Ala ma kota";

const txt2 = txt.slice(0,3);
console.log(txt2); //"Ala"

const txt3 = txt.slice(1,5);
console.log(txt3); //"la m"

const txt4 = txt.slice(4,6);
console.log(txt4); //"ma"

const txt5 = txt.slice(4);
console.log(txt5); //"ma kota"

const txt6 = txt.slice(-4);
console.log("Ala już nie ma " + txt6 + ", bo kocur jej zwiał..."); //Ala już nie ma kota, bo kocur jej zwiał...
```

Metoda **split(znak, długość)** zwraca tablicę, która składa się z podzielonych fragmentów tekstu. Miejsce podziału jest podawane w parametrze **znak**, a maksymalna ilość zwracanych elementów w parametrze **długość**:

```
const text = "Ala ma kota, a kot ma Alę, Ala go kocha, a Kot ją wcale ;("
const parts = text.split(", ");

// ["Ala ma kota", "a kot ma Alę", "Ala go kocha", "a Kot ją wcale ;("]
```

Metoda **replace(ciag_szukany, zamieniony)** zwraca nowy tekst, w którym został zamieniony szukany ciąg znaków na nowy tekst.

```
const text = "Ala ma kota"
const textNew = text.replace("kota", "psa");

console.log(textNew); //"Ala ma psa"
```