Többsoros Stringek és a `\n` Karakter

JavaScript-ben többsoros stringek létrehozásához használhatjuk a backtick (`) karaktereket, amelyeket sablon stringeknek is nevezünk. Ez lehetővé teszi, hogy egy string több sorban legyen, anélkül, hogy speciális karaktereket kellene használnunk. Azonban használhatjuk a `\n` karaktert is, amely új sort jelent a stringben.

Példa backtick karakterekkel:

```
let tobbSorosString = `Ez egy többsoros
string, amely
több sorban van.`;
console.log(tobbSorosString);
```

Példa `\n` karakterekkel:

```
let tobbSorosString = "Ez egy többsoros\nstring, amely\ntöbb sorban van.";
console.log(tobbSorosString);
```

Mindkét példa ugyanazt az eredményt adja, de a backtick karakterek használata általában olvashatóbb és karbantarthatóbb.

Stringek Indexelése

JavaScript-ben a stringek indexelése 0-tól kezdődik. Az egyes karakterekhez az indexükkel férhetünk hozzá.

Példa:

```
let szoveg = "JavaScript";
console.log(szoveg[0]); // "J"
console.log(szoveg[4]); // "S"
```

`toUpperCase` és `toLowerCase` Metódusok

A `toUpperCase` metódus egy stringet nagybetűssé alakít, a `toLowerCase` metódus pedig kisbetűssé.

Példa:

```
let szoveg = "JavaScript";
console.log(szoveg.toUpperCase()); // "JAVASCRIPT"
console.log(szoveg.toLowerCase()); // "javascript"
```

`indexOf` Metódus

Az `indexOf` metódus megkeresi egy adott alstring első előfordulását a stringben, és visszaadja annak indexét. Ha az alstring nem található, `-1`-et ad vissza.

Szintaxis:

```
string.indexOf(searchValue, fromIndex);
```

- `searchValue`: A keresendő alstring.
- `fromIndex`: Az index, amelytől kezdve a keresést elindítjuk (opcionális).

Példa:

```
let szoveg = "JavaScript";
console.log(szoveg.indexOf("Script")); // 4
console.log(szoveg.indexOf("Java")); // 0
console.log(szoveg.indexOf("Python")); // -1
```

`startsWith` Metódus

A `startsWith` metódus ellenőrzi, hogy egy string egy adott alstringgel kezdődik-e, és `true` vagy `false` értéket ad vissza.

Szintaxis:

```
string.startsWith(searchString, position);
```

- `searchString`: A keresendő alstring.
- `position`: Az index, amelytől kezdve a keresést elindítjuk (opcionális).

Példa:

```
let szoveg = "JavaScript";
console.log(szoveg.startsWith("Java")); // true
console.log(szoveg.startsWith("Script")); // false
console.log(szoveg.startsWith("Script", 4)); // true
```

`endsWith` Metódus

Az `endsWith` metódus ellenőrzi, hogy egy string egy adott alstringgel végződik-e, és `true` vagy `false` értéket ad vissza.

Szintaxis:

```
string.endsWith(searchString, length);
```

- `searchString`: A keresendő alstring.
- `length`: A string hossza, amelyig a keresést elvégezzük (opcionális).

Példa:

```
let szoveg = "JavaScript";
console.log(szoveg.endsWith("Script")); // true
console.log(szoveg.endsWith("Java")); // false
console.log(szoveg.endsWith("Java", 4)); // true
```

`slice` Metódus

A `slice` metódus egy új stringet hoz létre az eredeti string egy részéből, amely a megadott kezdő és záró index közötti karaktereket tartalmazza.

Szintaxis:

```
string.slice(start, end);
```

- `start`: A kezdő index (beleértve).
- `end`: A záró index (kizárva).

Példa:

```
let szoveg = "JavaScript";
console.log(szoveg.slice(0, 4)); // "Java"
console.log(szoveg.slice(4)); // "Script"
```

`substring` Metódus

A `substring` metódus egy új stringet hoz létre az eredeti string egy részéből, amely a megadott kezdő és záró index közötti karaktereket tartalmazza.

Szintaxis:

```
string.substring(start, end);
```

- `start`: A kezdő index (beleértve).
- `end`: A záró index (kizárva).

Példa:

```
let szoveg = "JavaScript";
console.log(szoveg.substring(0, 4)); // "Java"
console.log(szoveg.substring(4)); // "Script"
```

`substr` Metódus

A `substr` metódus egy új stringet hoz létre az eredeti string egy részéből, amely a megadott kezdő indexnél kezdődik, és a megadott hosszúságú.

Szintaxis:

```
string.substr(start, length);
```

- `start`: A kezdő index (beleértve).
- `length`: A kivágott rész hosszúsága.

Példa:

```
let szoveg = "JavaScript";
console.log(szoveg.substr(0, 4)); // "Java"
console.log(szoveg.substr(4, 6)); // "Script"
```

Stringek Összehasonlítása

A stringek összehasonlítására a ==, ===, < és > operátorokat használhatjuk. Az egyenlőség operátorok a stringek értékét hasonlítják össze, míg a < és > operátorok lexikografikus (ábécé) sorrendben hasonlítják össze őket.

Példa:

```
let szoveg1 = "apple";
let szoveg2 = "banana";
let szoveg3 = "apple";

console.log(szoveg1 == szoveg2); // false
console.log(szoveg1 === szoveg3); // true
console.log(szoveg1 < szoveg2); // true (mert "apple" előbb jön, mint "banana")
console.log(szoveg2 > szoveg3); // true (mert "banana" később jön, mint "apple")
```

`toLocaleCompare` Metódus

A `toLocaleCompare` metódus lehetővé teszi két string lokális összehasonlítását, tehát az ékezetes betűket helyesen kezelve és háromféle értéket ad vissza:

- `-1` ha az első string kisebb
- `1` ha az első string nagyobb
- `0` ha a két string egyenlő

Szintaxis:

```
string1.toLocaleCompare(string2);
```

Példa:

```
let szoveg1 = "apple";
let szoveg2 = "banana";
let szoveg3 = "apple";

console.log(szoveg1.toLocaleCompare(szoveg2)); // -1
console.log(szoveg2.toLocaleCompare(szoveg1)); // 1
console.log(szoveg1.toLocaleCompare(szoveg3)); // 0
```