Strings as Set of Characters

This lesson explains some of the methods that could be used to access or iterate over a string.

We'll cover the following

Identifying a Particular Character

You may think of a string as an array of characters. Each character is identified by a number called an index, just as it does for an array. The same golden rules apply:

- The index of the first character in a string is 0, not 1
- The highest index number is the string's length minus 1

Accessing a Particular Character

You know how to identify a character by its index. To access it, you use the *brackets notation* `[]` with the character index placed between the brackets.

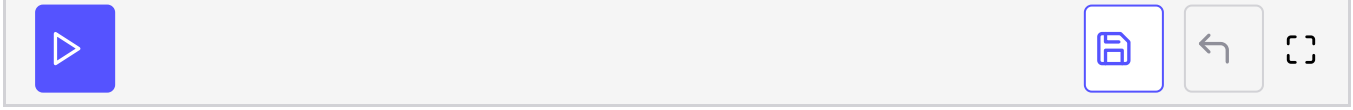
> Trying to access a string character beyond the string length produces an `undefined` result.

```
1  const sport = "basketball";
2  console.log(sport[0]);  // firs
3  console.log(sport[6]);  // sec
4  console.log(sport[10]); // und
```

Iterating Over a String

Now what if you want to access all string characters one-by-one? You could access each letter individually, as seen above:

```
const name = "Sarah"; // 5 characters
console.log(name[0]); // "S"
console.log(name[1]); // "a"
console.log(name[2]); // "r"
console.log(name[3]); // "a"
console.log(name[4]); // "h"
```

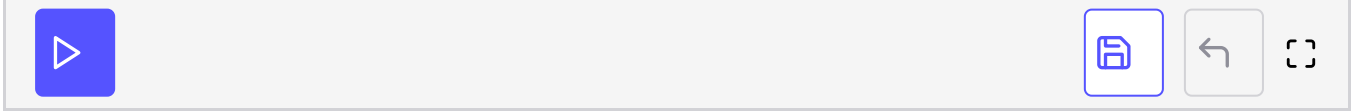


This is impractical if your string contains more than a few characters. You need a better solution to repeat access to characters. Does the word “repeat” bring to mind a former concept? Loops, of course! You may write a *loop* to access each character of a string. Generally speaking, a `for` loop is a better choice than a `while` loop, since we know the loop needs to run once for each character in the string.

```
for (let i = 0; i < myString.length; i++) {
    // Use myString[i] to access each character one by one
}
```

The loop counter `i` ranges from 0 (the index of the string’s first character) to string length - 1 (index of the last character). When the counter value equals the string length, the expression becomes false and the loop ends. So, the previous example may also be written with a `for` loop for an identical result.

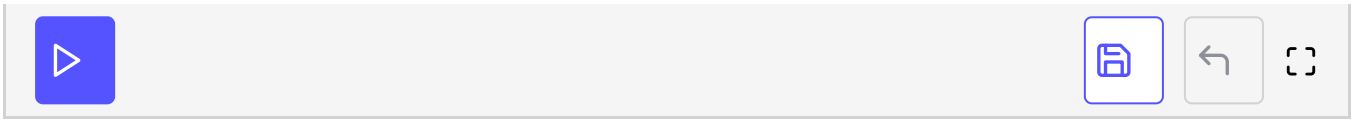
```
const name = "Sarah";
for (let i = 0; i < name.length; i++) {
  console.log(name[i]);
}
```



As for arrays covered earlier, a recent JavaScript evolution has introduced yet another option to iterate over a string: the `for...of` loop. The previous example

another option to iterate over a string: the `for-of` loop. The previous example may also be written:

```
const name = "Sarah";
for (const letter of name) {
  console.log(letter);
}
```



If the index is not needed inside the loop, this syntax is arguably simpler than a standard `for` loop.