string methods

String methods are actions that are called on strings.

string[index] With arrays, you get items by index. With strings you get *characters* by index.

1. Declare a string and get the first and third characters:

```
let fruit = 'cherry';
// get the first and third characters:
console.log(fruit[0]); // c
console.log(fruit[2]); // e
```

string.length

As with arrays, the length prpoerty of a string returns the number of items, in this case characters.

2. Get the length of a password and then run an if-statement that tells if the password is too short (less than 15 characters):

```
let password = '35khD%ewG@1';
let pswdLen = password.length;
console.log('Password length:', pswdLen);
if(password.length < 15) {
    console.log('Password is too short!');
}
```

string.replace(a, b)

The **replace()** method is called on a string. It takes two arguments:

- what you want to replace
- what you want to replace with

It returns a new string, without changing the original. Change 'cherry' to 'berry':

```
let newStr = fruit.replace('ch', 'b');
console.log(newStr); // berry
```

Change 'n' to 't' in an attempt to change 'banana' to 'batata' (a kind of sweet potato):

```
fruit = 'banana';
let foodItem = fruit.replace('n', 't');
console.log(foodItem); // batana
```

Notice that we only got 'batana', which means that the replace() method only changes the first occurrence of the target character(s).

The **replace()** method can replace a space with a word. Turn 'fresh salad' into 'fresh garden salad':

```
appetizer = appetizer.replace(' ', ' garden '));
console.log(appetizer); // fresh garden salad
```

replaceAll()

To replace all instances of a character, use replaceAll():

```
console.log(fruit); // banana
foodItem = fruit.replaceAll('n', 't');
console.log(foodItem); // batata
```

includes()

The **includes()** method takes a character(s) as its argument and returns true or false, based on whether or not the character(s) are in the string;

```
let bev = 'papaya smoothie';
console.log(bev.includes('smooth')); // true
console.log(bev.includes('mango')); // false
```

indexOf()

The **indexOf()** method takes a character(s) as its argument and returns the index of the first occurrence:

```
console.log(fruit); // banana
console.log(fruit.indexOf('n')); // 2
```

lastIndexOf()

The **lastIndexOf()** method takes a character(s) as its argument and returns the index of the last occurrence:

```
console.log(fruit); // banana
console.log(fruit.lastIndexOf('n')); // 4
```

If the char is not found in the string, indexOf returns -1:

```
console.log(fruit.indexOf('x')); // -1
```

The **indexOf()** method is useful for finding the space between two words, which gives the length of the first word:

```
let appetizer = 'fresh salad';
console.log(appetizer.indexOf(' ')); // 5
```

charAt()

The **charAt()** method is called on a string. It takes an integer as an argument and returns the character at that index:

```
let fruit = 'apple';
console.log(fruit.charAt(0)); // a
console.log('Hello World'.charAt(6)); // W
```

slice()

The **slice()** method returns a substring of the string it is called on, without changing the original string. It takes two arguments: a starting and an ending index. The end index is exclusive, meaning it is not included in the returned string. If the end index is omitted, it slices to the end.

```
let dynamicDuo = 'Batamn and Robin';
let hero1 = dynamicDuo.slice(0, 7);
console.log(hero1);
```

indexOf() and lastIndexOF() with slice()

But what if we didn't know the length of the word. We would have to get the index of the first space and then use that as the end index:

```
let superCouple = 'Superman and Lois Lane';
let indxSpace = superCouple.indexOf(' ');
let superhero = superCouple.slice(0, indxSpace);
console.log(superhero);
```

To get 'Lois Lane', we could get the index of the first 'L' and then just slice to the end of the string:

```
let Lindex = superCouple.indexOf('L');
let supermansGF = superCouple.slice(Lindex);
console.log(supermansGF);
```

To get the last word of a string, get the index of the last space, and then slice from there to the end:

```
let lastSpaceIndex = superCouple.lastIndexOf(' ');
let lastWord = superCouple.slice(lastSpaceIndex);
console.log(lastWord);
```

split()

The **split()** method is called on a string and returns an array.

```
let movieQuote = 'Show me the money';
var movieQuoteArr1 = movieQuote.split();
console.log(movieQuoteArr1); // ['Show me the money']
```

Notice that there is only one item in the resulting array. To have each word become an array item, pass the split() method the space between the words as its **delimiter**:

```
let movieQuoteArr2 = movieQuote1.split(' ');
console.log(movieQuoteArr2); // ['Show', 'me', 'the', 'money']
```

toLowerCase()

The **toLowerCase()** method is called on a string and returns an all-lowercase version of the string, without changing the original:

```
let newsFlash = 'Yankees Win World Series';
console.log(newsFlash.toLowerCase());
console.log(newsFlash); // unchanged
```

string.split() + array.join() + toLowerCase() Split and join can be used to break a string into an array and then put it back as a string, with some transformation being done in the process.

Turn this news headline into a hyphenated file name.

```

```

```
let news = 'Mets Win World Series';
let newsArr = headline.split(' ');
console.log(newsArr);
// ['Mets', 'Win', 'World', 'Series']
let imgFile = newsArr.join('-') + '.jpg';
console.log(imgFile);
// Mets-Win-World-Series.jpg;
let imgFileLC = imgFile.toLowerCase();
 console.log(imgFileLC);
// mets-win-world-series.jpg;
```

toUpperCase()

The **toUpperCase()** method is called on a string and returns an all-uppercase version of the string, without changing the original:

```
let headline = 'Jets Win Superbowl';
let shouting = headline.toUpperCase());
console.log(shouting); // JETS WIN SUPERBOWL
```

toUpperCase() + slice() to capitalize a word

There is no method for capitalizing a word, but we can get the first character and uppercase that. We also slice off the rest of the string, and then reconnect that part to the uppercased first letter.

Get the first letter and uppercase it:

```
let firstName = 'alexandria';
let firstChar = firstName[0];
let firstCharUC = firstChar.toUpperCase();
console.log(firstCharUC);
```

Get the rest of the string:

```
let restOfName = firstName.slice(1);
```

Connect the capitalized first letter with the rest of the string:

```
let capitalizedName = firstCharUC + restOfName;
console.log(capitalizedName);
```

NEXT: 04.05 Lab Exercises