Introduction to Strings

Let's do a deep dive into **strings**, one of the data types that your likely going to be working with most frequently.

String Variables Defined

A string is one or more text elements between quotes. So for example, if we were to create a variable called `letter`, and assign that the value of a capital "A", `letter` would be a string. If we created a variable called `sentence`, and assign it the value, "a sentence is a string", `sentence` is also a string variable.

String variables are treated typically as primitive variables. And a little later in this course we are going to be getting into the difference between primitive, and reference type variables.

String Primitives

Primitive variables pass their value when they're copied leaving the original value unchanged. This is important to understand because it's going to change the way that you interact with string variables.

```
var sentence = "A sentence is a string.";
var sentence2 = sentence;

console.log(sentence);
// "A sentence is a string."

console.log(sentence2);
// "A sentence is a string."

sentence2 = "This is a different sentence.";

console.log(sentence);
// "A sentence is a string."

console.log(sentence2);
// "This is a different sentence."
```

What you can see here is that `sentence` and `sentence2` are different variables. And the fact that you set `sentence2` to the value of `sentence`, doesn't mean that `sentence2` has any control over what happens to `sentence` when you make a change to `sentence2`.

That might seem like an intuitive result, but that's not always the case with variables in JavaScript, so pay attention.

Some String Methods and Properties

Once you start using string variables, the first thing you're going to want to do is start manipulating those strings. There are a number of methods and properties to know about for variables that are strings.

`Methods` are functions, and they manipulate the data inside of a variable, sometimes making changes to that variable, and sometimes not. Properties tell you information about the contents of that variable, without changing the contents. One of the properties of strings is the `length` telling how many characters there are between the quotes.

One of the methods on strings is `toUpperCase()`, which changes all the letters in a string to upper-case versions of themselves. You'll notice that methods are represented with parentheses at the end. If you don't include the parentheses, the method won't work. Another method on strings is `toLowerCase()`, which changes all of the letters to their lower-case versions. You can also use the `charAt()` method passing in a position to find out what letter is at what position in the string.

Examples of String Methods and Properties

```
var sentence = "A sentence is a string.";

console.log(sentence.length);
// 23

console.log(sentence.toUpperCase());
// "A SENTENCE IS A STRING."

console.log(sentence.toLowerCase());
// "a sentence is a string."

console.log(sentence.charAt(0));
// "A"

console.log(sentence.indexOf("is"));
// 11

console.log(sentence);
// "A sentence is a string.
```

As I mentioned before, none of these methods change the value of the original variable. They just take a copy of the contents, perform an operation on it, and report the result.

Examples of String Search Methods

Now let's discuss some methods that perform search in a string:

```
var sentence = "A sentence is a string.";

console.log(sentence.replace("sentence", "word"));
// "A word is a string."

console.log(sentence.substring(2, 10));
// "sentence"

console.log(sentence.substr(2, 8));
// "sentence"

console.log(sentence.substring(2));
// "sentence is a string."

console.log(sentence.substr(2));
// "sentence is a string."

console.log(sentence);
// "A sentence is a string."
```

`replace` searches for a word in a string and replaces it with another word.

`substring` takes a part of the string starting from the provided index. You can also pass a second argument to mark the ending index. If it is not provided, `substring` takes any characters from the start index till the end of the string.

These methods do not make any changes to the original variable. All they do is make a copy of the contents, and then perform manipulations in that copy, leaving the original untouched.