Lesson:

String

PW SKILLS

Topics

- Introduction to string
- Using different quotes for string
- Property access & its problems
- Escape characters
- String methods
- Template literals

Introduction to string

In JavaScript, the string is an object with a sequence of characters enclosed in single or double quotes used to represent and manipulate text, and it is one of the most commonly used datatypes in Javascript.
String is a primitive data type, which means that it is a fundamental type of data that cannot be broken down into smaller parts. String are immutable, which means that they cannot be changed once they are created.

Example

```
Unset
const myString = “hello”
```

String Properties - it includes length, which returns the number of characters present in the string.Example

```
JavaScript
console.log("hello".length); // 5
```

Using different quotes for string

In JavaScript, the string can be created by wrapping the text inside single quotes(‘ ’), double quotes (“ ”) or backticks(``)

Example

```
Unset
// Single quotes
const str = 'This is a string with single quotes';

// Double quotes
const str1 = "This is also a string with double quotes";

// Backticks
const str2 = `This is a backtick string.`;
```

Property access & its problems

In JavaScript, Property access in strings is a powerful feature that allows you to access and manipulate individual characters in a string. You can access individual characters within a string using an index. The index is a numerical value that represents the position of a character in the string.

Using Bracket Notation in string -
To access a specific character in a string, you can use square brackets [ ] with the index of the character you want to access.

Example

```
Unset
let myStr = 'I love Javascript';
console.log(myStr[0]);  // Output: 'I'
console.log(myStr[3]);  // Output: 'o'
```

However, Strings are immutable in JavaScript, meaning you can't modify them directly by index. You'll need to create a new string.
Example

```
Unset
let myStr = 'Hello, world!';
myStr[0] = '';  // This won't change the string
console.log(myStr);  // Output: 'Hello, world!'
```

Escape characters

String escape characters in JavaScript are used to represent special characters in a String. The backslash (\) character is used as an escape character which allows you to include those characters within a string that are difficult to type or represent directly.
Some of the escape characters are as follows -

- \' for a single quote
- \" for a double quote
- \\ for a backslash
- \n for a newline
- \t for a tab

Example -

```
Unset
// escape single quote
let singleQuote = 'This escape a single quote \' ';

// escape double quote
let doubleQuote = "This escape the double quote\" ";

// escape backslash
let backslash = 'This escape backslash: \\';

//  escape create newline
let newline = 'Line 1\nLine 2';

// escape to create tab space
let tab = 'Item\tDescription';
```

String methods

In JavaScript, string methods are functions that can be applied to string values to perform various operations or manipulations on those strings. These methods allow you to change the content of a string, extract information from a string, or perform transformations on a string.

Some of the commonly used String methods are as follows -

1. slice
2. substring
3. substr
4. replace
5. replaceAll
6. toUpperCase
7. toLowerCase
8. concat
9. split
10. indexOf
11. lastIndexOf
12. startsWith
13. endsWith
14. search
15. Trim
16. charAt
17. at
18. charCodeAt

1. slice - method extracts a section of a string and returns it as a new string, without modifying the original string.
Syntax - String.slice(separator) String.slice(separator, limit)

```javascript
// slice string method
console.log("Hello World!".slice(6)); // World!
console.log("Hello World!".slice(6, 11)); // World
```

2. substring - method returns the part of the string from the start index up to and excluding the end index
Syntax - String.subString(start), String.subString(start, end)

```javascript
// substring method
console.log("hello".substring(2)); // llo
console.log("hello".substring(1, 3)); // el
```

3. substr (deprecated) - method returns a portion of this string, starting at the specified index and extending for a given number of characters afterwards.
Syntax - String.substr(start) and String.substr(start, length)

```javascript
// subtstr string method (depricated)
const str = 'PW SKills';
console.log(str.substr(1, 2)); // w
console.log(str.substr(2)); // SKills
```

4. replace - method is used to replace a specified substring with another substring.
Syntax - String.replace(pattern, replacement)

```javascript
console.log("hello world world".replace("world", "earth"));
// output - hello earth
```

5. replaceAll - method returns a new string with all matches of a pattern replaced by a replacement.
Syntax - String.replaceAll(pattern, replacement)

```javascript
console.log("hello world world".replaceAll("world", "earth")); //
output - hello earth earth
```

6. toUpperCase - method returns the calling string value converted to lowercase
Syntax - String.toUpperCase()

```javascript
console.log("hello world".toUpperCase()); // HELLO WORLD
```

7. toLowerCase - method returns the calling string value converted to uppercase
Syntax - String.toLowerCase()

```javascript
console.log("Hello Word".toLowerCase()); // hello world
```

8. concat - this method concatenates (join) string.
Syntax - String.concat(), String.concat(str1), String.concat(str1, str2),

```javascript
const str1 = 'PW';
const str2 = 'SKILLS';
console.log(str1.concat(' ', str2));// PW SKILLS
console.log(str2.concat(', ', str1));// SKILLS, PW
```

9. split - method takes a pattern and divides a String into an ordered list of substrings by searching for the pattern, puts these substrings into an array, and returns the array.
Syntax - String.slice(separator) String.slice(separator,limit)

```javascript
console.log("hello,world,hello,world".split(","));
// output - [ 'hello', 'world', 'hello', 'world' ]
```

10. indexOf - this method searches for a string and returns its index position of the forest occurrence of the specified string. It returns -1 if the char is not available.
Syntax - String.indexOf(searching), String.indexOf(searchString, position)

```javascript
console.log("indexOf".indexOf("O")); // 5
console.log("indexOf".indexOf("m")); // -1
```

11. lastIndex - method of String values searches this string and returns the index of the last occurrence of the specified substring
Syntax - lastIndexOf(searchString), lastIndexOf(searchstring, position)

```JavaScript
const paragraph = "Join PW SKILL to enhance your skill"
const searchTerm = 'PW';

console.log(`The index of the first "${searchTerm}" from the end
is ${paragraph.lastIndexOf(searchTerm)}`);
// output: "The index of the first "PW" from the end is 5"
```

12. startsWith – method determines whether a string begins with the characters of a specified string, returning true or false as appropriate.
Syntax - String.startsWith(searchString), String.startsWith(searchString)

```JavaScript
console.log("hello World".startsWith("hello")); // true
```

13. endsWith – his method determines whether the string ends with the characters of a specified string. Returning true or false.
Syntax - String.endsWith (searchString), String.endsWith (searchString)

```JavaScript
console.log("endswith".endsWith("with")); // true
console.log("endswith".endsWith("end")); // false
```

14. search – method of String values executes a search for a match between a regular expression and this string, returning the index of the first match in the string.
Syntax - String.search(searchValue)

```JavaScript
let str = 'Hello, PW! Welcome to the world.';
let searchTerm = 'PW';
let result = str.search(searchTerm);
console.log('Search term:', searchTerm); // Search term: PW
console.log('Result:', result);  //Result: 7
```

15. trim – method removes whitespace from both ends of a string and returns a new string, without modifying the original string.
Syntax - String.trim()

```javascript
console.log("  hello world    ".trim());
// output - hello world
```

16. chartAt – it takes an integer value and returns the string located with the corresponding integer located in it.
Syntax - String.charAt(index)

```javascript
console.log("chartAt".charAt(3)); // r
```

17. at – it takes an integer value and returns a new string, it allows positive and negative integers. Negative integers count back from the last string character.
Syntax - String.at(index)

```javascript
console.log("hello".at(1)); // e
console.log("hello".at(-1)); // o
```

18. charCodeAt – method of String values returns an integer between 0 and 65535 representing the UTF-16 code unit at the given index.
Syntax - String.charCodeAt(index)

```javascript
const str = 'PW SKILLS';
const index = 4;
console.log("char code", str.charCodeAt(index));
```

Template literals

In JavaScript, Template literals are a new way to create strings that were introduced in ES6. They are a more flexible way to create string allowing interpolation of variables and expressions. They are enclosed in backticks (``) instead of single or double quotes.

Some of the uses of Template literals -
1. Multi-line strings - the Template literals can be used to create multi-lne without having to use the **\n** escape character
For example –

```JavaScript
const myString = `Hello template literals.
It can span multiple lines without having to use the \n escape character.`;

console.log(myString);

// output ----------
Hello template literals.
It can span multiple lines without having to use the
 escape character.
```

2. String interpolation – it is the process of inserting expressions such as variables, function calls, and arithmetic expressions, into a string. It is a powerful feature that can make code more concise and readable

To perform string interpolation in JavaScript, you use template literals. Template literals are strings that are enclosed in backticks (`) instead of single quotes (') or double quotes ("). Within a template literal, you can use placeholders to represent expressions. Placeholders are enclosed in curly braces ({}) prefix with dollar symbol ($).

For example –

```Unset
let name = 'Mang';
let age = 22;

const greeting = `Hello, my name is ${name}, I am ${age} years old.`;

console.log(greeting);  // Output: Hello, my name is Mang, I am 22 years old.
```

The string interpolation also allows us to create a multi-line string
Example –

```Unset
const notify = `
There was an error processing your request.

Please try again later.
`;

console.log(notify);
// - - - Output - - -

There was an error processing your request.

Please try again later.
```