Extracing string parts:

→ slice(start, end)
→ substring(start, end)
→ substr(start, length)

~~Javastring~~

Javascript string slice()

slice() extracts a part of a string and returns the extracted part in a new string.

→ The method takes 2 parameters
  → start position
  → end position.

```
let text = "Apple, Banana, Kiwi";
                0 1 2 3 4 5 6 7
let part = text.slice(7,13);
document.getElementById("demo").
    innerHTML = part;
</script>
```

o/p:- Banana

→ Javascript counts position from zero.
→ First position is 0.
→ Second position is 1.

```
let text = "Apple, Banana, Kiwi";
                0 1 2 3 4 5
let part = text.slice(7)
document.getElementById("demo").inner
    HTML = part;
</script>
```

o/p:- Banana, Kiwi

→ If a parameter is negative, the position is counted from the end of string

```
let text = "Apple, Banana, Kiwi";
let part = text.slice(-12);
```

o/p:- Banana, Kiwi

→ let text = "Apple, Banana, Kiwi";
let part = str.substr(7, 6);
substr() is similar to slice().
The difference is that the second parameter specifies the length of the extracted part.

o/p:- Banana

→ If you omit the second parameter, substr() will slice out the rest of string.

let str = "Apple, Banana, Kiwi";
let part = str.substr(7);

Banana, Kiwi

→ If the first parameter is negative, the position counts from the end of the string

let str = "Apple, Banana, Kiwi";
let part = str.substr(-4);

Kiwi

→ Replacing string content:
replace() method replaces a specified value with another value in a string.

let text = "please visit Microsoft";
let newText = text.replace("Microsoft",
"W3Schools");

→ the replace() method does not change the string it is called on.
→ The replace method returns a new string
→ The replace() method replaces only the first match.
→ If you want to replace all matches, use a regular expression with the /g flag etc.

let text = "please visit Microsoft and
Microsoft!";
let newText = text.replace(/Microsoft/,
"W3Schools");

By default the replace() method is case sensitive.

```
let text = "Please visit Microsoft!";
let newText = text.replace("Microsoft", "W3Schools");
```

→ To replace case insensitive, use a regular expression with an /i flag (insensitive).

```
let text = "Please visit Microsoft!";
let newText = text.replace(/MICROSOFT/i, "W3Schools");
```

→ To replace all matches, use a regular expression with a /g flag (global match).

```
let text = "Please visit Microsoft and Microsoft!";
let newText = text.replace(/Microsoft/g, "W3Schools");
```

→ JavaScript String replaceAll()

```
text = text.replaceAll("cats", "dogs");
text = text.replaceAll("Cats", "dogs");
```

```
let text = "I love cats. Cats are very easy to love. Cats are very popular.
I love dogs. Dogs are very easy to love. Dogs are very popular.
```

→ The replaceAll() method allows you to specify a regular expression instead of string to be replaced.

→ If the parameters is a regular expression the global flag (g) must be set, otherwise typeerror is thrown.

```
text = text.replaceAll(/cats/g, "dogs");
text = text.replaceAll(/cats/g, "dogs");
```

Converting to upper and lower:

→ A string is converted to upper case with toUpperCase().

```
let text1 = "Hello world";
let text2 = text1.touppercase();
```

→
```
let text1 = "Hello world";
let text2 = text1.tolowercase();
```

String concatenation:-

```
<!DOCTYPE html>
<html>
<body>
<h1>Javascript string</h1>
<h2> The concat() Method </h2>
<p> the concat() method joins two or more
    strings</p>
<p id="demo"></p>
<script>
let text1 = "Hello";
let text2 = "world";
let text3 = text1.concat(" ", text2);
document.getElementById("demo").innerHTML
    = text3;
</script>
</body>
</html>
```

```
text = "Hello" + " " + "world";
text = "Hello".concat(" ", "world");
```

→ All string methods return a new string. They don't modify the original string.

→ Strings are immutable:- strings cannot be changed only replaced.

Javascript String trim():-

→ trim() method removes whitespace from both sides of a string.

```
let text1 = "   Hello world!   ";
```

```
<!DOCTYPE html>
<html>
<body>
<h1> Javascipt string </h1>
<h2> the concat() method </h2>
<p> the concat() method joins two or more
      strings </p>
<p id="demo"></p>
<script>
let text1="Hello";
let text2="world";
let text3=text1.concat(" ",text2);
document.getElementById("demo").innerHTML
    =text3;
</script>
</body>
</html>
```

Javascipt string trimstart():-

the trimstart() method works like trim(), but removes whitespaces only start of a string.

```
let text1="  Hello world!  ";
let text2=text1.trimstart();
```

trimEnd():-

→ It removes whitespaces at end of string

```
<!DOCTYPE html>
<html>
<body>
<h1> Javascript strings </h1>
<h2> the trimEnd() Method </h2>
<p id="demo"></p>
<script>
let text1="Hello world!  ";
let text2=text1.trimEnd();
document.getElementById("demo").inner
    HTML="length text1= "+text1.
    length+"<br> length text2 =" +
```

</body>
</html>

length text1 = 22
length text2 = 17

JavaScript string Padding:-

padStart()
PadEnd()

JavaScript string padStart():-

```
<!DOCTYPE html>
<html>
<body>
<h1> JavaScript strings</h1>
<h2> the padStart() method </h2>
<p> the padStart() method pads a string
from the start. </p>
<p id="demo"></p>
<script>
let text="5";
text = text.padStart(4,"0");
document.getElementById("demo").innerHTML
        =text;
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<h1> Javascript strings </h1>
<h2> The padStart() Method</h2>
<p id="demo"></p>
<script>
let text="5";
document.getElementById("demo").innerHTML
```

</body>
</html>

⇒ xxx5.

→ The padStart5() method is a string method.
→ to pad a number, convert the number to a string first.

```
→ <!DOCTYPE html>
  <html>
  <body>
  <h1> JavaScript string 5</h1>
  <p id="demo" 5</p>
  <script>
   let numb= 5;
   let text= numb. toString5();
  document. getElementById("demo"). innerHTML
    =text. padStart(4,0);
  </script>
   </body>
  </html>
```

⇒0005.

JavaScript string padEnd()

The padEnd() method pads a string from the end. It pads a string with another string until it reaches a given length.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
let text="5";
```

</html>

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
let text = "5";
document.getElementById("demo").innerHTML
    = text.padEnd(4,"x");
</script>
</body>
</html>
```

o/p :- 5xxx

→ to pad a number method is a string method.
→ To pad a number, convert the number to a string first.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
let numb = 5;
let text = numb.toString();
document.getElementById("demo").innerHTML
    = text.padEnd(4,"x");
</script>
</body>
</html>
```

⇒ 5xxx.

Extracting String Characters:

→ charAt(position)
→ charCodeAt(position)
→ property access() → [] method is a new

```
Const name = "W3schools";
let letter = name.at(2);
```

o/p :- s.

</script>
</body>
</html> ⇒ 5000

② <!DOCTYPE html>
<html>
<body>

<script>
let text = "5";
document.getElementById("demo").innerHTML
= text.padEnd(4,"x");
</script>
</body>
</html>

o/p:- 5xxx

→ to pad a number method is a string method.
→ To pad a number, convert the number to a string first.

<!DOCTYPE html>
<html>
<body>

<script>
let numb = 5;
let text = numb.toString();
document.getElementById("demo").innerHTML
= text.padEnd(4,"x");
</script>
</body>
</html> ⇒ 5xxx.

Extracting String Characters:

→ charAt(position)
→ charCodeAt(position)
→ property access()
→ At(position)

Const name = "w3schools"
let letter = name.at(2);
o/p → s.

→ At method is a new addition to JavaScript.
→ It allows negative indexe while charAt() do no

charAt() method returns the character at a specified index (poistion) in a string.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var text="HELLO WORLD";
document.getElementById("demo").innerHTML
    =text.charAt(0);
</script>
</body>
</html>
```

=> H.

Javascript String charCodeAt()

It returns the unicode of the character at a specified text in a string:-

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
let text="HELLO WORLD";
document.getElementById("demo").innerHT
    ML=text.charCodeAt(0);
</script>
</body>
</html>
```

=> 72

Property Access:-

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
```

```
let text = "HELLO WORLD";
document.getElementById("demo").inner
    HTML = text[0];
</script>
</body>
</html>
```

=> H.

→ property access might be a little unpredictable.
→ It makes strings look like arrays (but they are not).
→ If no character is found, [] returns undefined, while charAt() returns an empty string.
→ It is read only. str[0] = "A" gives no error (but does not work)

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
  let text = "HELLO WORLD";
  text[0] = "A";
  document.getElementById("demo").innerHTML
      = text;
</script>
</body>
</html>
```

=> HELLO WORLD

Converting a string to an Array:-

If you want to work with a string as an array you can convert it to an array.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
let text = "a,b,c,d,ef";
text.split(",");
```

```
getElementById("demo").inne
HTML = text[0];
</script>
</body>
</html>
```

⇒ H.

→ property access might be a little unpredictable.
→ It makes strings look like arrays (but they are not).
→ If no characters is found, [] returns undefined, while charAt() returns an empty string.
→ It is read only. str[0] = "A" gives no error (but does not work)

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
let text = "HELLO WORLD";
text[0] = "A";
document.getElementById("demo").innerHTML
= text;
</script>
</body>
</html>
```

⇒ HELLO WORLD

Converting a string to an Array:-

If you want to work with a string as an array you can convert it to an array.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
let text = "a, b, c, d, ef";
```

innerHTML = myArray;
</script> </body>
</html>

=>a

→ If the separator is omitted, the returned array will contain the whole string in index[0].
→ If the separator is "", the returned array will be an array of single characters.

text.split("")

Javascript String Search:-

String indexOf()
String lastIndexOf()
String search()
String match()
String matchAll()
String includes()
String startsWith()
String endsWith()

Javascript String index of():-

→ indexOf() method returns the index (position) of first occurence of a string in a string:-

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
let text = "please locate where 'locate' occurs!";
let index = text.indexOf("locate");
document.getElementById("demo").innerHTML = index;
</script>
```