HINDUSTAN
INSTITUTE OF TECHNOLOGY & SCIENCE
(DEEMED TO BE UNIVERSITY)

CSB4301 - WEB TECHNOLOGY

B.Tech – V Semester

UNIT II

Dr. Muthukumaran M
Associate Professor
School of Computing Sciences,
Department of Computer Science and Engineering

Strings & Manipulations

String Length

The length property returns the length of a string:

```
let txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
txt.length     // Returns 26
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Properties</h2>

<p>The length property returns the length of a string:</p>

<p id="demo"></p>

<script>
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
document.getElementById("demo").innerHTML = text.length;
</script>

</body>
</html>
```

Extracting String Parts

There are 3 methods for extracting a part of a string:

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

The slice() Method

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

The method takes 2 parameters: the start position, and the end position (end not included).

This example slices out a portion of a string from position 7 to position 12 (13-1):

```
let str = "Apple, Banana, Kiwi";
str.slice(7, 13)     // Returns Banana
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The slice() method extract a part of a string
and returns the extracted parts in a new string:</p>

<p id="demo"></p>

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML =
str.slice(7,13);
</script>

</body>
</html>
```

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

This example slices out a portion of a string from position -12 to position -6:

```
let str = "Apple, Banana, Kiwi";
str.slice(-12, -6)    // Returns Banana
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The slice() method extract a part of a string
and returns the extracted parts in a new string:</p>

<p id="demo"></p>

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.slice(-12,-
6);
</script>

</body>
</html>
```

If you omit the second parameter, the method will slice out the rest of the string:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The slice() method extract a part of a string
and returns the extracted parts in a new string:</p>

<p id="demo"></p>

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.slice(7);
</script>

</body>
</html>
```

or, counting from the end:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The slice() method extract a part of a string
and returns the extracted parts in a new string:</p>

<p id="demo"></p>

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.slice(-12);
</script>

</body>
</html>
```

The substring() Method

substring() is similar to slice().

The difference is that substring() cannot accept negative indexes.

```
let str = "Apple, Banana, Kiwi";
substring(7, 13)    // Returns Banana
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The substring() method extract a part of a string and
returns the extracted parts in a new string:</p>

<p id="demo"></p>

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML =
str.substring(7,13);
</script>

</body>
</html>
```

The substr() Method

substr() is similar to slice().

The difference is that the second parameter specifies the length of the extracted part.

```
let str = "Apple, Banana, Kiwi";
str.substr(7, 6)    // Returns Banana
```

The result of res will be:

Banana

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

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The substr() method extract a part of a string
and returns the extracted parts in a new string:</p>

<p id="demo"></p>

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.substr(7);
</script>

</body>
</html>
```

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

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The substr() method extract a part of a string
and returns the extracted parts in a new string:</p>

<p id="demo"></p>

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.substr(-4);
</script>

</body>
</html>
```

Converting to Upper and Lower Case

A string is converted to upper case with toUpperCase():

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>Convert string to upper case:</p>
<button onclick="myFunction()">Try it</button>
<p id="demo">Hello World!</p>
<script>
function myFunction() {
  let text = document.getElementById("demo").innerHTML;
  document.getElementById("demo").innerHTML =
  text.toUpperCase();
}
</script>
</body>
</html>
```

A string is converted to lower case with toLowerCase():

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>Convert string to lower case:</p>
<button onclick="myFunction()">Try it</button>
<p id="demo">Hello World!</p>
<script>
function myFunction() {
  let text = document.getElementById("demo").innerHTML;
  document.getElementById("demo").innerHTML =
  text.toLowerCase();
}
</script>

</body>
</html>
```

The concat() Method

concat() joins two or more strings:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</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>
```

The concat() method can be used instead of the plus operator. These two lines do the same:

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

String.trim()

The trim() method removes whitespace from both sides of a string:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String.trim()</h2>
<p>Click the button to alert the string with removed
whitespace.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
  let text = "     Hello World!     ";
  alert(text.trim());
}
</script>

</body>
</html>
```

JavaScript String Padding

ECMAScript 2017 added two String methods: padStart and padEnd to support padding at the beginning and at the end of a string.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>The padStart() method pads a string with another
string:</p>
<p id="demo"></p>
<script>
let text = "5";
document.getElementById("demo").innerHTML =
text.padStart(4,0);
</script>

</body>
</html>
```

Extracting String Characters

There are 3 methods for extracting string characters:

- charAt(position)
- charCodeAt(position)
- Property access [ ]

The charAt() Method

The charAt() method returns the character at a specified index (position) in a string:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>The charAt() method returns the character at a given
position in a string:</p>
<p id="demo"></p>

<script>
var text = "HELLO WORLD";
document.getElementById("demo").innerHTML =
text.charAt(0);
</script>

</body>
</html>
```

The charCodeAt() Method

The charCodeAt() method returns the unicode of the character at a specified index in a string:

The method returns a UTF-16 code (an integer between 0 and 65535).

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>The charCodeAt() method returns the unicode of the
character at a given position in a string:</p>
<p id="demo"></p>
<script>
let text = "HELLO WORLD";
document.getElementById("demo").innerHTML =
text.charCodeAt(0);
</script>
</body>
</html>
```

Property Access

ECMAScript 5 (2009) allows property access [ ] on strings:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>ECMAScript 5 allows property access on strings:</p>

<p id="demo"></p>

<script>
var str = "HELLO WORLD";
document.getElementById("demo").innerHTML = str[0];
</script>
</body>
</html>
```

Converting a String to an Array

A string can be converted to an array with the split() method:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>
<p>Display the first array element, after a string split:</p>

<p id="demo"></p>

<script>
let text = "a,b,c,d,e,f";
const myArray = text.split(",");
document.getElementById("demo").innerHTML = myArray[0];
</script>

</body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>Using String.split():</p>
<p id="demo"></p>
<script>
let text = "Hello";
const myArr = text.split("");
text = "";
for (let i = 0; i < myArr.length; i++) {
  text += myArr[i] + "<br>"
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

JavaScript String Search

JavaScript methods for searching strings:

String.indexOf()
String.lastIndexOf()
String.startsWith()
String.endsWith()

String.indexOf()

The indexOf() method returns the index of (the position of) the first occurrence of a specified text in a string:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>

<p>The indexOf() method returns the position of the first
occurrence of a specified text:</p>
<p id="demo"></p>
<script>
let str = "Please locate where 'locate' occurs!";
document.getElementById("demo").innerHTML =
str.indexOf("locate");
</script>
</body>
</html>
```

String.lastIndexOf()

The lastIndexOf() method returns the index of the last occurrence of a specified text in a string:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The lastIndexOf() method returns the position of the last
occurrence of a specified text:</p>

<p id="demo"></p>

<script>
let str = "Please locate where 'locate' occurs!";
document.getElementById("demo").innerHTML =
str.lastIndexOf("locate");
</script>

</body>
</html>
```

Both indexOf(), and lastIndexOf() return -1 if the text is not found:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>Both indexOf(), and lastIndexOf() return -1 if the text is not found:</p>

<p id="demo"></p>

<script>
let str = "Please locate where 'locate' occurs!";
document.getElementById("demo").innerHTML = str.indexOf("John");
</script>

</body>
</html>
```

String.search()

The search() method searches a string for a specified value and returns the position of the match:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The search() method returns the position of the first
occurrence of a specified text in a string:</p>

<p id="demo"></p>

<script>
let str = "Please locate where 'locate' occurs!";
document.getElementById("demo").innerHTML =
str.search("locate");
</script>

</body>
</html>
```

The two methods, indexOf() and search(), are equal?

They accept the same arguments (parameters), and return the same value?

The two methods are NOT equal. These are the differences:

The search() method cannot take a second start position argument.

The indexOf() method cannot take powerful search values (regular expressions).

String.match()

The match() method searches a string for a match against a regular expression, and returns the matches, as an Array object.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Search</h2>

<p>Search a string for "ain":</p>

<p id="demo"></p>
<script>
let text = "The rain in SPAIN stays mainly in the plain";
document.getElementById("demo").innerHTML =
text.match(/ain/g);
</script>
</body>
</html>
```

String.includes()

The includes() method returns true if a string contains a specified value.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Search</h2>

<p>Check if a string includes "world":</p>
<p id="demo"></p>
<p>The includes() method is not supported in Internet
Explorer.</p>
<script>
let text = "Hello world, welcome to the universe.";
document.getElementById("demo").innerHTML =
text.includes("world");
</script>
</body>
</html>
```

String.startsWith()

The startsWith() method returns true if a string begins with a specified value, otherwise false:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Strings</h2>

<p>Check if a string starts with "Hello":</p>
<p id="demo"></p>
<p>The startsWith() method is not supported in Internet
Explorer.</p>
<script>
let text = "Hello world, welcome to the universe.";
document.getElementById("demo").innerHTML =
text.startsWith("Hello");
</script>
</body>
</html>
```

String.endsWith()

The endsWith() method returns true if a string ends with a specified value, otherwise false:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Strings</h2>

<p>Check if a string ends with "Doe":</p>

<p id="demo"></p>
<p>The endsWith() method is not supported in Internet
Explorer.</p>
<script>
let text = "John Doe";
document.getElementById("demo").innerHTML =
text.endsWith("Doe");
</script>
</body>
</html>
```