JavaScript String Methods

- description : JavaScript String Methods
- author : 오션
- email : shlim@repia.com
- lastupdate : 2021-04-23

The source of this article

JavaScript String Methods
문자열 메서드는 문자열 작업에 도움이 됩니다.

String Methods and Properties

“John Doe”와 같은 원시 값(Primitive values)은 속성이나 메서드를 가질 수 없습니다 (객체가 아니기 때문에).

그러나 JavaScript에서는 메서드와 속성을 실행할 때 원시 값을 객체로 취급하기 때문에,
JavaScript로, 메서드와 속성도 원시 값에 사용할 수 있습니다.

String Length

length 속성은 문자열의 길이를 반환합니다.

```
<!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 txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    let sln = txt.length;
    document.getElementById("demo").innerHTML = sln;
  </script>
</body>
</html>
```

Finding a String in a String

indexOf() 메서드는 문자열에서 지정된 텍스트의 처음 나타나는 곳(occurrence)의 인덱스(위치)를 반환합니다.

```
<!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!";
    let pos = str.indexOf("locate");
    document.getElementById("demo").innerHTML = pos;  // 7
  </script>
</body>
</html>
```

%%JavaScript%%는 0부터 위치를 계산합니다.
0은 문자열의 첫 번째 위치, 1은 두 번째, 2는 세 번째 ...

lastIndexOf() 메서드는 문자열에서 지정된 텍스트가 마지막으로 나타나는 곳의 시작 위치(인덱스)를 반환합니다.

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript String Methods</h2>
  <p>The lastIndexOf() method returns the positon of the last occurrence of a specified text:</p>
  <p id="demo"></p>
  <script>
    let str = "Please locate where 'locate' occurs!";
    let pos = str.lastIndexOf('locate');
    document.getElementById("demo").innerHTML = pos;  // 21
  </script>
</body>
</html>
```

텍스트가 발견되지 않는 경우, indexOf() 및 lastIndexOf()는 모두 -1을 반환합니다.

```
<!DOCTYPE html>
```

```
<html>
<body>
  <h2>JavaScript String Methods</h2>
  <p>Both indexOf() and lastIndexOf() returns -1 if the text is not
found:</p>
  <p id="demo"></p>
  <script>
    let str = "Please locate where 'locate' occurs!";
    let pos = str.indexOf("John");
    document.getElementById("demo").innerHTML = pos;  // -1
  </script>
</body>
</html>
```

두 메서드 모두 검색의 시작 위치를 두 번째 매개 변수로 허용합니다.

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript String Methods</h2>
  <p>The indexOf() method accepts a second parameter as thee starting
position for the search:</p>
  <p id="demo"></p>
  <script>
    let str = "Please locate where 'locate' occurs!";
    let pos = str.indexOf("locate", 15);  //index[15]부터 검색을 시작
    document.getElementById("demo").innerHTML = pos;  // 21
  </script>
</body>
</html>
```

`lastIndexOf()` 메서드는 뒤로(끝에서 처음으로) 검색합니다.
즉, 두 번째 매개 변수가 15이면, 검색은 위치 15에서 시작하고, 문자열의 시작 부분까지 검색합니다.

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript String Methods</h2>
  <p>The lastIndexOf() method accepts a second parameter as the stating
position for the search.</p>
  <p>Remember that the lastIndexOf() method searches backwords, so position
15 means start the search at position 15, and search to the beginning.</p>
  <p>Position 15 is position 15 from the beginning</p>
  <p id="demo"></p>
  <script>
    let str = "Please locate where 'locate' occurs!";
    let pos = str.lastIndexOf("locate", 15);
    document.getElementById("demo").innerHTML = pos;  // 7
    console.log(str.length);  // 36
```

```
    console.log(str[15]);     // h
  </script>
</body>
</html>
```

Tip:lastIndexOf() 메서드는 지정된 인덱스 값에서 왼쪽(index[0]의 방향으로)으로 검색을 시작하면서 처음으로 나타나는 searchValue의 원래 인덱스 번호를 반환한다.

Searching for a String in a String

search() 메서드는 지정된 값에 해당하는 문자열을 검색하고, 일치하는 위치를 반환합니다.

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript String Methods</h2>
  <p>The search() method returns the position of the first occurrnece of a specified text in a string.</p>
  <p id="demo"></p>
  <script>
    let str = "Please locate where 'locate' occurs!";
    let pos = str.search("locate");
    document.getElementById("demo").innerHTML = pos;  // 7
  </script>
</body>
</html>
```

Did You Notice?

두 메서드 indexOf() 및 search()는 동일합니까?

동일한 인수(매개 변수)를 수용하고 동일한 값을 반환합니까?

이 두 가지 메서드는 동일하지 않습니다. 차이점은 다음과 같습니다.

- search() 메서드는 두 번째 시작 위치 인수를 사용할 수 없습니다.
- indexOf() 메서드는 강력한 검색 값 (정규식)을 사용할 수 없습니다.

이후 장에서 정규식에 대해 자세히 알아볼 것입니다.

Extracting String Parts

문자열의 일부분을 추출하는 3가지 방법이 있습니다.

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

The slice() Method

slice()는 문자열의 일부를 추출하고, 추출된 부분을 새 문자열로 반환합니다.

이 메서드는 시작 위치와 끝 위치 (끝이 포함되지 않음)라는 두 가지 매개 변수를 사용합니다.

다음 예제는 위치 7에서 위치 12 (13-1)까지 문자열의 일부를 자릅니다.

```
<!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";
    let res = str.slice(7, 13); // 7-시작, 13-종료, 포함X
    document.getElementById("demo").innerHTML = res;    // Banana
  </script>
</body>
</html>
```

기억하세요 : 자바 스크립트는 0부터 위치를 계산합니다. 첫 번째 위치는 0입니다.

매개 변수가 음수이면, 위치는 문자열의 끝부터 계산됩니다.

다음 예제는 -12 위치에서 -6 위치까지 문자열의 일부를 잘라냅니다.

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

추가 설명 from MDN:

문법

```
str.slice(beginImdex[, endIndex])
```

```
beginIndex:
- 입력값이 추출 시작점이 되는 인덱스
- 음수인 경우, beginIndex = str.length(문자열 길이) + beginIndex
- 에제) beginIndex = -3 인 경우, 시작점은 strLength + (-3)
endIndex:
- 0부터 시작하는 추출 종료점 인덱스로 그 직전까지 추출
-  endIndex 위치의 문자는 추출에 포함되지 않습니다.
```

두 번째 매개변수가 없는 경우, 메서드는 해당 문자열의 나머지 부분을 잘라냅니다.
(– > 입력된 매개변수가 나타내는 인덱스 번호의 지점부터 인덱스 마지막 번호까지의 문자열을 잘라냅니다)

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

두 번째 매개변수가 없으면서, 첫 번째 매개변수가 음수인 경우는 마지막 인텍스에서부터 계산합니다.

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

```
Internet Explorer 8 및 이전 버전에서는 음수 위치가 작동하지 않습니다.
```

The substring() Method

substring()은 slice()와 유사합니다.

차이점은 substring()은 음수 인덱스를 허용 할 수 없다는 것입니다.

```
  <script>
    let str = "Apple, Banana, Kiwi";
    let res = str.substring(7, 13);    // index7부터 12까지 추출
    document.getElementById("demo").innerHTML = res;  // Banana
  </script>
```

두 번째 매개변수를 생략하면 substring()은 나머지 문자열을 잘라냅니다.

The substr() Method

substr()은 slice()와 유사합니다.

차이점은 두 번째 매개변수가 추출될 부분의 **길이**를 지정한다는 것입니다.

```
<script>
  let str = "Apple, Banana, Kiwi";
  let res = str.substr(7, 10);   // index7부터 10개에 해당하는 문자를 추출
  document.getElementById("demo").innerHTML = res;  // Banana, Ki
</script>
```

두 번째 매개변수를 생략하면, substr()은 나머지 문자열을 잘라냅니다.

```
<script>
  let str = "Apple, Banana, Kiwi";
  let res = str.substr(8);   // index7부터 끝까지의 문자을 추출
  document.getElementById("demo").innerHTML = res;  // anana, Kiwi
</script>
```

첫 번째 매개 변수가 음수이면 위치는 문자열 끝부터 계산된 인덱스 지점부터 오른쪽으로 나머지를 추출합니다.

```
<script>
  let str = "Apple, Banana, Kiwi";
  let res = str.substr(-8);   // index-8부터 끝까지의 문자을 추출
  document.getElementById("demo").innerHTML = res;  // na, Kiwi
</script>
```

Replacing String Content

replace() 메서드는 문자열에서 지정된 값을 다른 값으로 대체(교환)합니다.

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript String Methods</h2>
  <p>Replace "Microsoft" with "W3Schools" in the paragraph below:</p>
  <button onclick="myFunction()">Try it</button>
  <p id="demo">Please visit Microsoft!</p>
  <script>
    function myFunction() {
      let str = document.getElementById("demo").innerHTML;
      let txt = str.replace("Microsoft", "W3Schools");
      document.getElementById("demo").innerHTML = txt;
    }
```

```
  </script>
</body>
</html>
```

''replace()'' 메서드는 호출되는 문자열을 변경하지 않습니다. 새 문자열을 반환합니다.

기본적으로, replace() 메서드는 **첫 번째 일치 항목만** 교환합니다.

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript String Methods</h2>
  <p>Replace "Microsoft" with "W3Schools" in the paragraph below:</p>
  <button onclick="myFunction()">Try it</button>
  <p id="demo">Please visit Microsoft and Microsoft!</p>
  <script>
    function myFunction() {
      let str = document.getElementById("demo").innerHTML;
      let txt = str.replace("Microsoft", "W3Schools");
      document.getElementById("demo").innerHTML = txt;
    }
  </script>
</body>
</html>
```

기본적으로 replace() 메서드는 대소 문자를 구분합니다.
대문자로 작성한 MICROSOFT는 작동하지 않습니다.

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript String Methods</h2>
  <p>Try to replace "Microsoft" with "W3Schools" in the paragraph below:</p>
  <button onclick="myFunction()">Try it</button>
  <p id="demo">Please visit Microsoft!</p>
  <script>
    function myFunction() {
      let str = document.getElementById("demo").innerHTML;
      let txt = str.replace("MICROSOFT", "W3Schools");
      document.getElementById("demo").innerHTML = txt;
    }
  </script>
  <P><strong>Note:</strong> Nothing will happen. By default, the replace()
method is case sensitive. Writing MICROSOFT (with upper-case) will not
```

```
work.</P>
</body>
</html>
```

대소문자 구분없이 교체하기 위해서, /i 플래그(대소문자 비구분)와 함께 정규식을 사용하십시오.

```
<script>
  function myFunction() {
    let str = document.getElementById("demo").innerHTML;
    let txt = str.replace(/MICROSOFT/i, "W3Schools");
    document.getElementById("demo").innerHTML = txt;
  }
</script>
```

정규식은 따옴표 없이 작성됩니다.

모든 일치 항목을 바꾸려면, **정규식**을 /g 플래그 (전역 일치)와 함께 사용하십시오.

```
<script>
  function myFunction() {
    let str = document.getElementById("demo").innerHTML;
    let txt = str.replace(/Microsoft/g, "W3Schools");
    document.getElementById("demo").innerHTML = txt;
  }
</script>
```

global match & case-insensitive

```
<script>
  function myFunction() {
    let str = document.getElementById("demo").innerHTML;
    let txt = str.replace(/MiCROSOFT/gi, "W3Schools");
    document.getElementById("demo").innerHTML = txt;
  }
</script>
```

Converting to Upper and Lower Case

문자열은 toUpperCase()를 사용하여 대문자로 변환됩니다.

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

```
  <p>Convert string to upper case:</p>
  <button onclick="myFunction()">Try it</button>
  <p id="demo">Bonjour le Monde</p>
  <script>
    function myFunction() {
      let text = document.getElementById("demo").innerHTML;
      document.getElementById("demo").innerHTML = text.toUpperCase();
    }
  </script>
</body>
</html>
```

toLowerCase()를 사용하여 문자열을 소문자로 변환합니다.

```
  <script>
    function myFunction() {
      let text = document.getElementById("demo").innerHTML;
      document.getElementById("demo").innerHTML = text.toLowerCase();
    }
  </script>
```

The concat() Method

concat()은 두 개 이상의 문자열을 결합합니다.

```
<script>
    let text1 = "Impossible is";
    let text2 = "Nothing";
    let text3 = text1.concat(" ", text2);  // // text1+공백+text2
    document.getElementById("demo").innerHTML = text3;  // Impossible is
Nothing
  </script>
```

더하기 연산자 대신 concat() 메서드를 사용할 수 있습니다. 아래의 두 줄은 동일합니다.

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

모든 문자열 메서드는 새 문자열을 반환합니다. 원래 문자열을 수정하지 않습니다.
공식적으로 말하면: 문자열은 불변(immutable)입니다: 문자열은 변경될 수 없고, 대체만 가능합니다.

String.trim()

trim() 메서드는 문자열의 양쪽에 있는 공백을 제거합니다.

```
<script>
  function myFunction() {
    let str = "           Hello World!        ";
    alert(str.trim());
  }
</script>
```

Internet Explorer 8 이하에서는 trim () 메서드가 지원되지 않습니다.

IE 8을 지원해야 하는 경우, 대신 정규식과 함께 replace()를 사용할 수 있습니다.

```
<script>
  let str = "      Hello World!     ";
  alert(str.replace(/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g, ''));
</script>
```

위의 replace 해결책을 사용하여 JavaScript String.prototype에 트림 메서드를 추가할 수도 있습니다.

```
```

JavaScript String Padding

ECMAScript 2017은 두 가지 String 메서드인 padStart 및 padEnd를 추가하여 문자열의 시작과 끝에서 패딩을 지원합니다.

```
```

```
```

Internet Explorer에서는 문자열 패딩이 지원되지 않습니다.

Firefox와 Safari는 JavaScript 문자열 패딩을 지원하는 최초의 브라우저였습니다.

Extracting String Characters

문자열 문자를 추출하는 방법에는 세 가지가 있습니다.

- charAt(position)
- charCodeAt(position)
- 속성 액세스 []

The charAt() Method

charAt() 메서드는 문자열의 지정된 인덱스(위치)에 있는 문자를 반환합니다.

The charCodeAt() Method

charCodeAt() 메서드는 문자열의 지정된 인덱스에 있는 문자의 유니코드를 반환합니다.

이 메서드는 UTF-16 코드 (0에서 65535 사이의 정수)를 반환합니다.

Property Access

ECMAScript 5 (2009)는 문자열에 대한 속성 액세스[]를 허용합니다.

속성 액세스는 약간 예측 불가능할 수 있습니다.
* Internet Explorer 7 또는 이전 버전에서는 작동하지 않습니다.
* 문자열을 배열처럼 보이게 합니다 (그러나 그렇지 않습니다).
* 문자가 없으면, %%[]%%는 %%undefined%%를 반환하고, %%charAt()%%은 빈 문자열을 반환합니다.
* 읽기 전용입니다. %%str[0] = "A"%%는 오류를 제공하지 않지만 작동하지 않습니다!

문자열을 배열로 사용하려면, 문자열을 배열로 변환 할 수 있습니다.

Converting a String to an Array

`split()` 메서드를 사용하여 문자열을 배열로 변환 할 수 있습니다.

```
```

구분자(separator)를 생략하면, 반환된 배열에는 인덱스 [0]에 전체 문자열이 포함됩니다.\\ Fix Me!
구분자 기호가 ""인 경우, 반환된 배열은 단일 문자의 배열이 됩니다.

```
```

Complete String Reference

전체 참조를 보려면, Complete JavaScript String Reference로 이동하십시오.

이 참조에는 모든 문자열 속성 및 메서드에 대한 설명과 예제가 포함되어 있습니다.

오션, Javascript, String, Methods