

Topic: String methods

Strings

collection of words or characters enclosed in a single or double quotes

Strings are immutable and can't be changed directly. We can get the values of string but we can't change.

```
//iterate the string using for loop
var a = "hello";
for(i=0; i<5; i++){
  console.log(a[i]);
}
```

String.length

It is a method used to find the length of a string

The .length property returns the number of characters in the string, including spaces, punctuation marks, and special characters.

```
let str = "Hello, world!";
console.log(str.length); // This will output 13
```

charAt()

The charAt() method is used to return the character at a specified index (position) within a string.

```
let str = "Hello";
console.log(str.charAt(0)); // Output: "H"
console.log(str.charAt(1)); // Output: "e"
console.log(str.charAt(4)); // Output: "o"
```

at()

The at() method allows you to directly access a character at a specific position within a string, similar to charAt() but also takes negative values.

```
let str = "Hello";
console.log(str.at(0)); // Output: "H"
console.log(str.at(1)); // Output: "e"
console.log(str.at(-1)); // Output: "o"
```

charCodeAt()

the charCodeAt() method returns the Unicode value (integer between 0 and 65535) of the character at a specified index in a string.

```
let str = "Hello";
console.log(str.charCodeAt(0)); // Output: 72
console.log(str.charCodeAt(1)); // Output: 101
console.log(str.charCodeAt(4)); // Output: 111
```

slice(start,end)

The `slice()` method is used to extract a section of a string and return it as a new string. It doesn't modify the original string. This method takes two parameters: the start index and the end index (optional).

```
let str = "Hello, world!";
console.log(str.slice(0, 5)); // Output: "Hello"
console.log(str.slice(7)); // Output: "world!"
console.log(str.slice(-6)); // Output: "world!"
console.log(str.slice(7, -1)); // Output: "world"
console.log(str.slice(0)); // Output: "Hello, world!"
console.log(str.slice(-1)); // Output: "!"
```

substring(start, end)

The `substring()` method is used to extract a portion of a string and return it as a new string. It is similar to the `slice()` method, but there are differences in how negative indices are handled because negative values are considered as zero

```
let str = "Hello, world!";
console.log(str.substring(0, 5)); // Output: "Hello"
console.log(str.substring(7)); // Output: "world!"
console.log(str.substring(7, 12)); // Output: "world"
console.log(str.substring(-6)); // Output: "Hello, world!"
console.log(str.substring(7, -1)); // Output: "Hello, world"
```

substr()

In JavaScript, the `substr()` method is used to extract a portion of a string, starting from a specified index and extending for a specified length of characters. This method is different from `substring()` in that the second parameter specifies the length of the extracted substring rather than the end index.

```
let str = "Hello, world!";
console.log(str.substr(0, 5)); // Output: "Hello"
console.log(str.substr(7)); // Output: "world!"
console.log(str.substr(7, 5)); // Output: "world"
console.log(str.substr(-6)); // Output: "world!"
console.log(str.substr(7, -1)); // Output: ""
```

toUpperCase()

The `toUpperCase()` method is used to convert all characters in a string to uppercase letters.

```
let str = "Hello, world!";
let a = str.toUpperCase();
console.log(a); // Output: "HELLO, WORLD!"
```

toLowerCase()

The `toLowerCase()` method is used to convert all characters in a string to lowercase letters.

```
let str = "Hello, WORLD!";
let a = str.toLowerCase();
console.log(a); // Output: "hello, world!"
```

concat()

The `concat()` method is used to concatenate two or more strings.

```
let str1 = "Hello";
let str2 = "world";
let str3 = "!";
let result = str1.concat(" ", str2, str3);
console.log(result); // Output: "Hello, world!"
```

trim()

The `trim()` method is used to remove whitespace from both ends of a string.

```
let str = " Hello, world! ";
let trimmedStr = str.trim();
console.log(trimmedStr); // Output: "Hello, world!"
```

repeat()

The `repeat()` method is used to construct and return a new string by concatenating the string on which it is called a certain number of times.

```
let str = "Hello";
let repeatedStr = str.repeat(3);
console.log(repeatedStr); // Output: "HelloHelloHello"
```

Split()

The `split()` method is used to split a string into an array.

```
let str = "Hello, world!";
let parts = str.split(", ");
console.log(parts); // Output: ["Hello", "world!"]

let characters = str.split("");
console.log(characters); // Output: ["H", "e", "l", "l", "o", ",", " ", "w", "o", "r", "l", "d", "!"]
```

replace()

`replace()` method used to replace the current occurrences of substring within a string with another string

syntax: `string.replace(searchValue, replaceValue)`

```
let originalString = "Hello, world!,world";
let newString = originalString.replace("world", "universe");
console.log(newString); // Output: Hello, universe!,world
```

Above program replaces only the first match to replace all matches use a regular expression with `/g` flagset

```
let newString = originalString.replace(/world/g, "red fox");
```

```
let originalString = "Hello, world!,world";
let newString = originalString.replace(/world/g, "universe");
console.log(newString); // Output: Hello, universe!,universe
```

Replace method is case sensitive writing World will not consider

To replace case insensitive, use a regular expression with an /i

```
let originalString = "Hello, world!, World";
let newString = originalString.replace(/world/ig, "universe");
console.log(newString); // Output: Hello, universe!,universe
```

replaceAll()

it is a method to replace a substring with another string but it doesn't compatible in all browsers

string search methods

indexOf() and lastIndexOf()

```
let str = "Hello, world!";
let newString = str.indexOf("w");//output 5 because it checks from the starting
let newString1 = str.lastIndexOf("w");//output 8 because it checks from the ending
```

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

```
let newString1 = str.lastIndexOf("w",5)//output 8 because it checks from the ending
```

if the second parameter is 5, the search starts at position 5, and searches

search()

the search() method is used to search for a specified substring within a string. It returns the index of the first occurrence of the specified substring, or -1 if the substring is not found. It can take regular expressions also

```
let str = "Hello, world!";
let index = str.search("world");
console.log(index); // Output: 7
```

two methods, indexOf() and search(), are not equal because search() method cannot take a second start position argument.

match()

The match() method in JavaScript is used to search a string for a specified pattern (regular expression), and returns an array containing the matches, or null if no matches are found. It can take regular expression and it can print the values in an array

```
let text = "The rain in SPAIN stays mainly in the plain";
text.match(/ain/gi); //4 [ain,AIN,ain,ain]
```

includes()

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

```
let text = "Hello world, welcome to the universe.";
text.includes("world");//true
```

Template literals

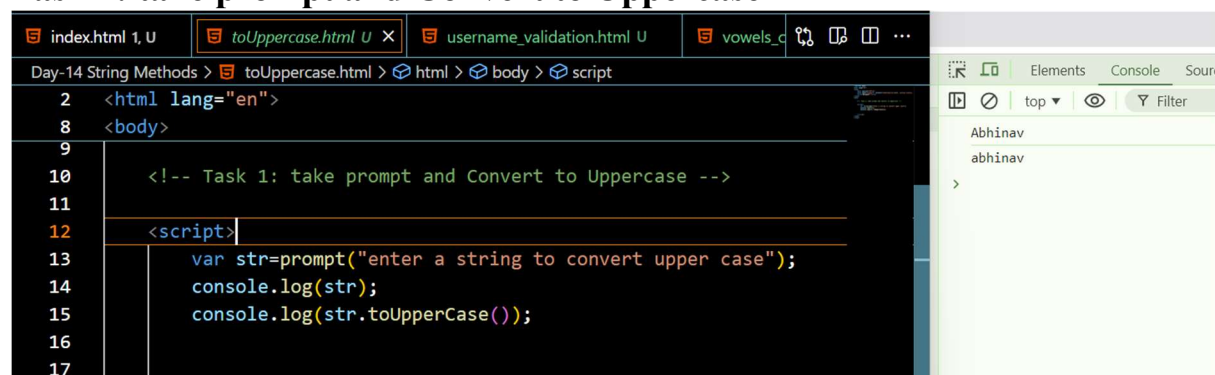
Template literals allow you to embed expressions and variables directly within the string using ``{}``. This makes string interpolation more intuitive and readable.

```
let a = 5;
let b = 10;
let result = `The sum of ${a} and ${b} is ${a + b}.`;
// result is "The sum of 5 and 10 is 15."
```

TASKS:

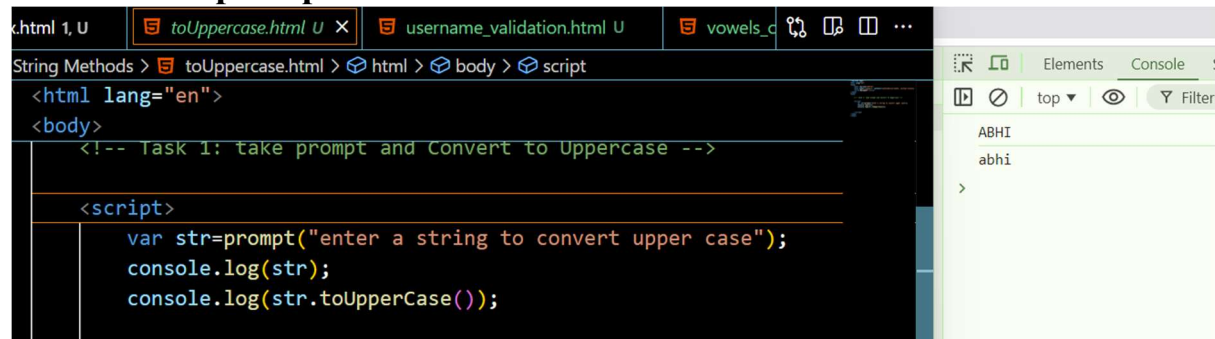
Basic level:

Task 1: take prompt and Convert to Uppercase



```
index.html 1, U | toUppercase.html U X | username_validation.html U | vowels_c | ...
Day-14 String Methods > toUppercase.html > html > body > script
2 <html lang="en">
8 <body>
9
10 <!-- Task 1: take prompt and Convert to Uppercase -->
11
12 <script>
13   var str=prompt("enter a string to convert upper case");
14   console.log(str);
15   console.log(str.toUpperCase());
16
17
```

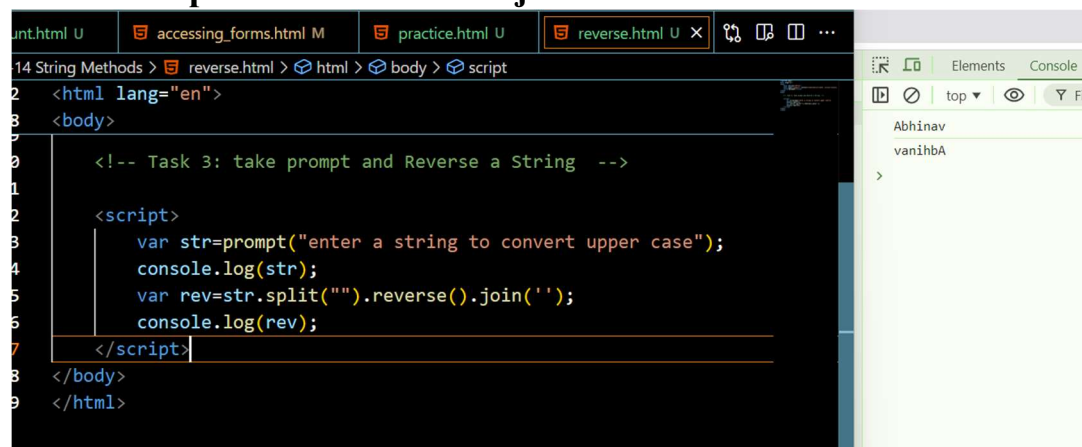
Task 2: take prompt and Convert to Lowercase



```
html 1, U | toUppercase.html U X | username_validation.html U | vowels_c | ...
String Methods > toUppercase.html > html > body > script
<html lang="en">
<body>
<!-- Task 1: take prompt and Convert to Uppercase -->
<script>
var str=prompt("enter a string to convert upper case");
console.log(str);
console.log(str.toUpperCase());
```

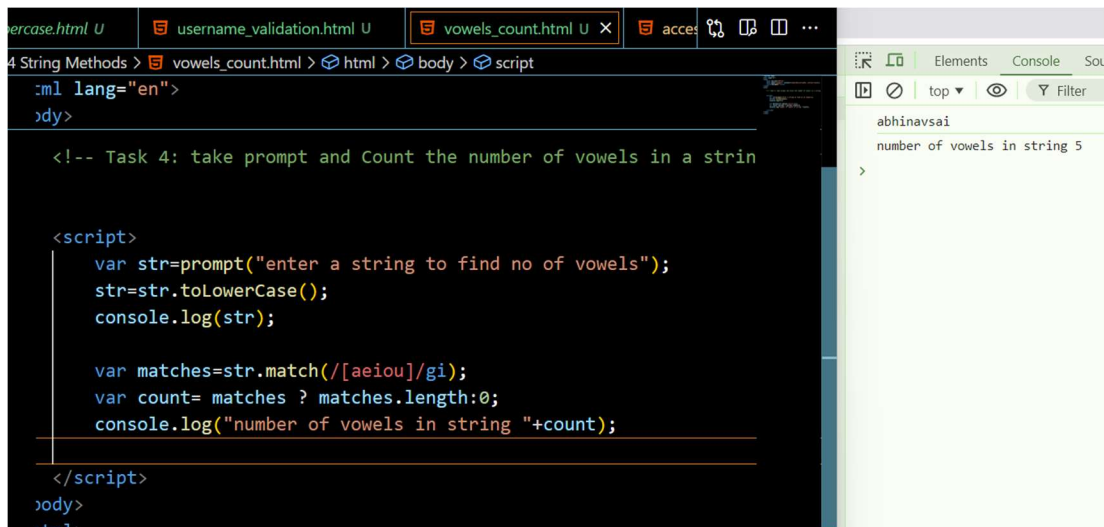
Task 3: take prompt and Reverse a String

hints: use split and reverse and join method



```
unth.html U | accessing_forms.html M | practice.html U | reverse.html U X | ...
14 String Methods > reverse.html > html > body > script
2 <html lang="en">
3 <body>
4
5 <!-- Task 3: take prompt and Reverse a String -->
6
7 <script>
8   var str=prompt("enter a string to convert upper case");
9   console.log(str);
10  var rev=str.split("").reverse().join('');
11  console.log(rev);
12 </script>
13 </body>
14 </html>
```

Task 4: take prompt and Count the number of vowels in a string-- hints use match method



The screenshot shows a web browser with a single tab titled 'vowels_count.html'. The browser's developer tools are open, showing the 'script' tab. The code in the script tag is as follows:

```
<!-- Task 4: take prompt and Count the number of vowels in a string -->

<script>
  var str=prompt("enter a string to find no of vowels");
  str=str.toLowerCase();
  console.log(str);

  var matches=str.match(/[aeiou]/gi);
  var count= matches ? matches.length:0;
  console.log("number of vowels in string "+count);
</script>
</body>
</html>
```

The console output shows the input string 'abhinavsai' and the resulting count 'number of vowels in string 5'.

Advanced

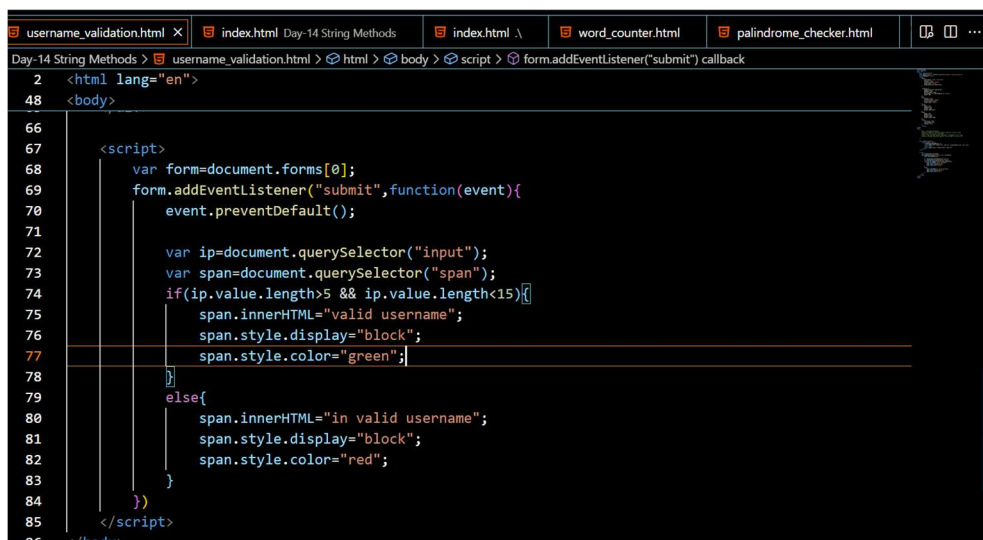
Task 1: Username Validation

Scenario: Validate a username based on specific criteria. Task:

Prompt the user to enter a username.

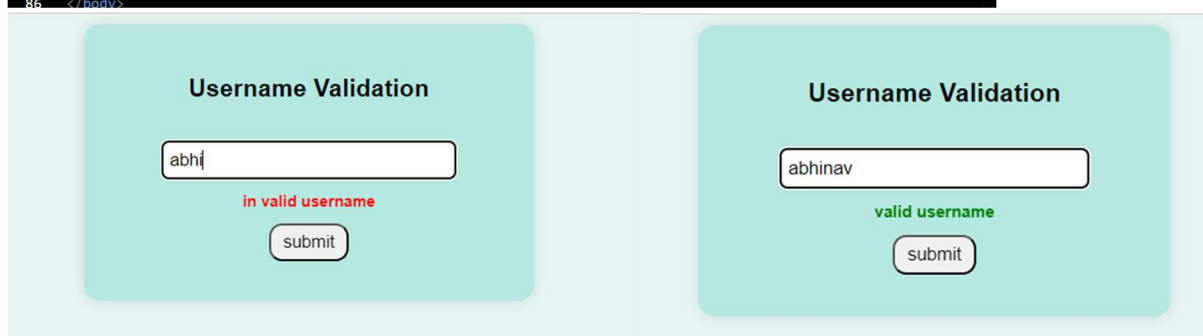
Check if the username contains characters in between 5 to 15 long.

Display a message indicating whether the username is valid or not.



The screenshot shows a web browser with multiple tabs. The active tab is 'username_validation.html'. The browser's developer tools are open, showing the 'script' tab. The code in the script tag is as follows:

```
2 <html lang="en">
48 <body>
66
67   <script>
68     var form=document.forms[0];
69     form.addEventListener("submit",function(event){
70       event.preventDefault();
71
72       var ip=document.querySelector("input");
73       var span=document.querySelector("span");
74       if(ip.value.length>5 && ip.value.length<15){
75         span.innerHTML="valid username";
76         span.style.display="block";
77         span.style.color="green";
78       }
79       else{
80         span.innerHTML="in valid username";
81         span.style.display="block";
82         span.style.color="red";
83       }
84     })
85   </script>
86 </body>
```



The image shows two screenshots of a 'Username Validation' form. The form has a text input field and a 'submit' button. In the first screenshot, the input field contains 'abhi' and the text 'in valid username' is displayed in red below the input field. In the second screenshot, the input field contains 'abhinav' and the text 'valid username' is displayed in green below the input field.

Task 2: Email Formatter

Scenario: Format an email address. Task:

Prompt the user to enter their first and last name.

Convert the names to lowercase and concatenate them with a dot in between.

concat "@gmail.com" to the result and display the formatted email address.

inp: jenny and joy

out: jenny.joy@gmail.com

```
username_validation.html | email_formatter.html M X | index.html Day-14 String Methods | index.html \ | word_counter.html
Day-14 String Methods > email_formatter.html > html > body > div.container > form#fm > input#user.text
2  <html lang="en">
48  <body>
59      <div class="container">
60          <form action="" id="fm">
61              <h3>Email Formater</h3>
62              <input type="text" class="text" id="user" placeholder="Enter First name" >
63              <input type="text" class="text" id="user" placeholder="Enter Last name" >
64              <span></span>
65              <input type="submit" value="submit" id="sub">
66              <input type="text" class="text" id="user" placeholder="your genrated email will display he
67          </form>
68      </div>
69      <script>
70          var form=document.forms[0];
71          form.addEventListener("submit",function(event){
72              event.preventDefault();
73
74              var ip1=document.querySelectorAll("input")[0];
75              var ip2=document.querySelectorAll("input")[1];
76
77              res=`${ip1.value}.${ip2.value}@gmail.com`;
78              var ip3=document.querySelectorAll("input")[3];
79              ip3.setAttribute("value",res);
80          })
81      </script>
```

Email Formater

abhi

1252

submit

abhi.1252@gmail.com

Task 3: Word Counter

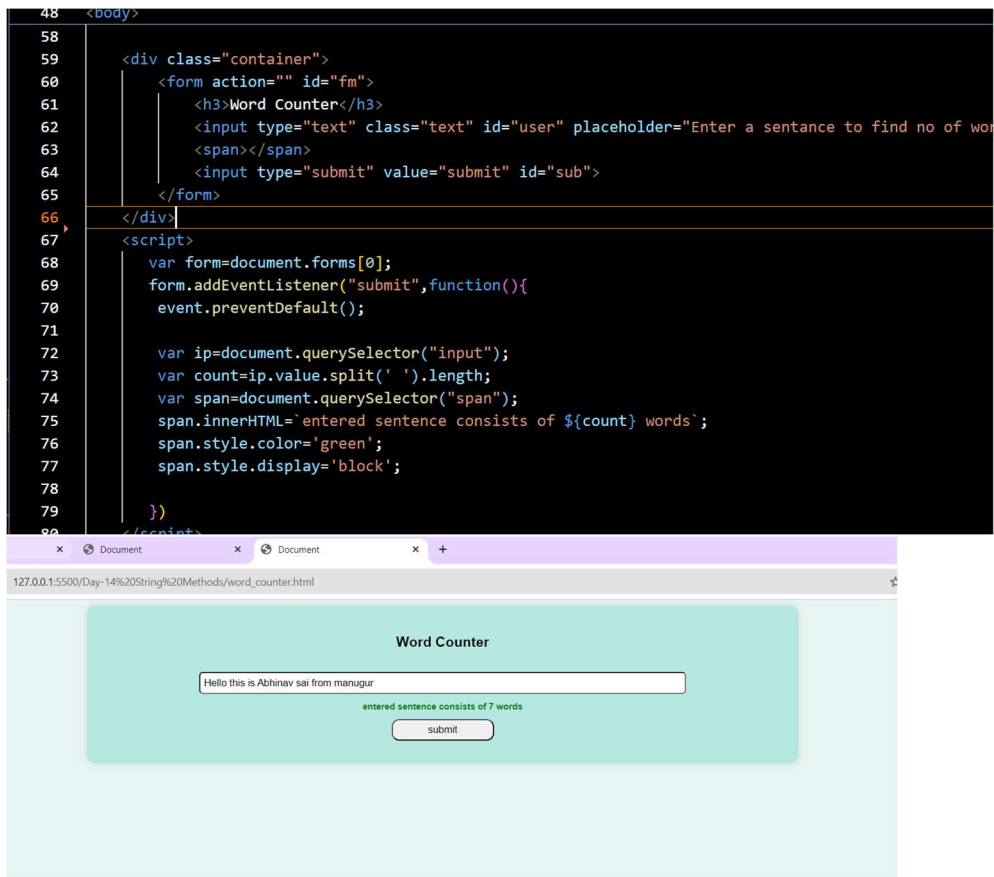
Scenario: Count the number of words in a sentence. Task:

Prompt the user to enter a sentence.

Split the sentence into words and count them.

Display the word count.

Hint: use split and length method



Task 4: Palindrome Checker

Scenario: Check if a given string is a palindrome. Task:

Prompt the user to enter a string.

Check if the string reads the same forwards and backwards.

Display a message indicating whether the string is a palindrome

Hints : use split,join, reverse method in arrays

inp:john

out: not a palandrome

inp:dad

out:it is a palandrome


```
<script>
  var form=document.forms[0];
  form.addEventListener("submit",function(){
    event.preventDefault();

    var ip=document.querySelector("input");
    var rev=ip.value.split('').reverse().join('');
    var span=document.querySelector("span");

    if(ip.value == rev){
      span.innerHTML="given word is palindrome";
      span.style.color='green';
      span.style.display='block';
    }
    else{
      span.innerHTML="given word is not a palindrome";
      span.style.color='red';
      span.style.display='block';
    }
  })
</script>
```

Palindrome Checker

given word is not a palindrome

Palindrome Checker

given word is palindrome