Day 4/Question02

1.Replace() Method of String :

The replace() method replaces a specified value with another value in a string:

Example:

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

The replace() method does not change the string it is called on. It returns a new string.

By default, the replace() method replaces only the first match:

Example:

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

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

By default, the replace() method is case sensitive. Writing PHONE (with upper-case) will not work:

Example:

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

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

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

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>Replace "Phone" with "Contact" in the paragraph below:</p>
<button onclick="myFunction()">Try it</button>
<p id="demo">Please write your Phone number!</p>
<script>
function myFunction() {
```

```
  var str = document.getElementById("demo").innerHTML;
  var txt = str.replace(/PHONE/i,"Contact");
  document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

2. To Replace All:

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

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>Replace all occurrences of "Phone" with "Contact" in the paragraph below:</p>
<button onclick="myFunction()">Try it</button>
<p id="demo">Please write your Phone number and Your Phone number must start with country code !</p>
<script>
function myFunction() {
  var str = document.getElementById("demo").innerHTML;
  var txt = str.replace(/Phone/g,"Contact");
  document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

3. Splice() Method of an Array:

The splice() method can be used to add new items to an array:

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Array Methods</h2>
<h2>splice()</h2>
<p>The splice() method adds new elements to an array.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = "Original Array:<br>" + fruits;
function myFunction() {
  fruits.splice(2, 0, "Lemon", "Kiwi");
  document.getElementById("demo2").innerHTML = "New Array:<br>" + fruits;
}
</script>
</body>
</html>
```

The first parameter (2) defines the position **where** new elements should be **added** (spliced in).
The second parameter (0) defines **how many** elements should be **removed**.
The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be **added**.

The splice() method returns an array with the deleted items:

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Array Methods</h2>
<h2>splice()</h2>
<p>The splice() method adds new elements to an array, and returns an array with the deleted elements (if any).</p>
<button onclick="myFunction()">Try it</button>
<p id="demo1"></p>
<p id="demo2"></p>
```

```
<p id="demo3"></p>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = "Original Array:<br> " + fruits;
function myFunction() {
  var removed = fruits.splice(2, 2, "Lemon", "Kiwi");
  document.getElementById("demo2").innerHTML = "New Array:<br>" + fruits;
  document.getElementById("demo3").innerHTML = "Removed Items:<br> " + removed;
}
</script>
</body>
</html>
```

Using splice() to Remove Elements:

With clever parameter setting, you can use splice() to remove elements without leaving "holes" in the array:

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Array Methods</h2>
<h2>splice()</h2>
<p>The splice() methods can be used to remove array elements.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;
function myFunction() {
  fruits.splice(0, 1);
  document.getElementById("demo").innerHTML = fruits;
}
</script>
</body>
</html>
```

The first parameter (0) defines the position where new elements should be added (spliced in).
The second parameter (1) defines how many elements should be removed**.**
The rest of the parameters are omitted. No new elements will be added.

4. substring() Method

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

Example:

```
<!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>
var str = "Apple, Banana, Kiwi";
var res = str.substring(7,13);
document.getElementById("demo").innerHTML = res;
</script>
</body>
</html>
```

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