1. What is Regular expression in JavaScript?

A regular expression, often abbreviated as "regex" or "regexp," is a sequence of characters that define a search pattern. In JavaScript, regular expressions are objects that allow you to perform pattern matching and search operations within strings.

Regular expressions can be used for various tasks such as:

- Searching for patterns within a string.
- Validating input to ensure it matches a desired format.
- Replacing parts of a string based on a pattern.
- Extracting specific information from strings.

In JavaScript, regular expressions can be created using the RegExp constructor or by using a literal syntax enclosed between slashes (/).

test(string): This method tests whether a string contains a match for the regular expression pattern. It returns **true** if a match is found, otherwise **false**.

```
var regex = /pattern/;
var str = "Some text containing pattern";
console.log(regex.test(str));

```

```
node /tmp/V9ZtWNWimi.js
true
```

exec(string): This method searches for a match within a string and returns an array containing information about the match. If no match is found, it returns **null**.

```
var regex = /pattern/;
var str = "Some text containing pattern";
console.log(regex.exec(str));

```

```
node /tmp/sL7kYnktPk.js
[
  'pattern',
  index: 21,
  input: 'Some text containing pattern',
  groups: undefined
]
```

match(string): This method is available on strings and works similarly to **exec()**. It searches for a match within the string and returns an array containing information about the match, or **null** if no match is found.

```
var str = "Some text containing pattern";
console.log(str.match(/pattern/));

```

```
node /tmp/3Q4C8TYhTw.js
[
  'pattern',
  index: 21,
  input: 'Some text containing pattern',
  groups: undefined
]
```

search(string): This method searches for a match within a string and returns the index of the first occurrence of the match, or **-1** if no match is found.

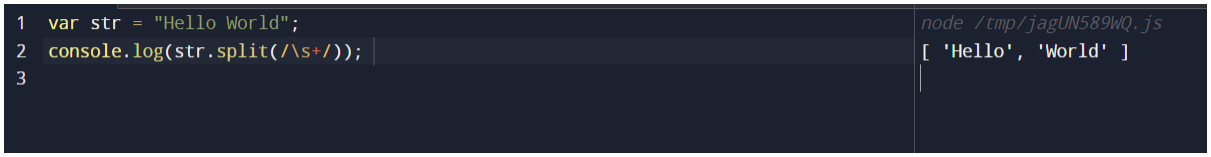
```
var str = "Some text containing pattern";
console.log(str.search(/pattern/));

```

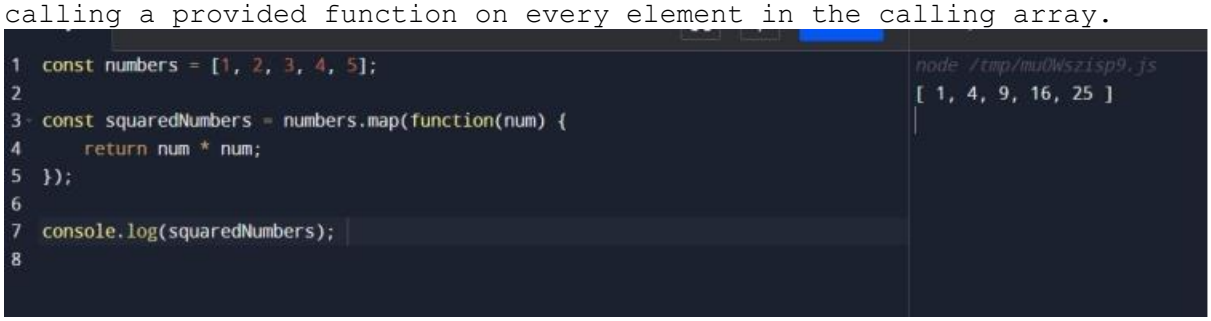
```
node /tmp/29COydD9j8.js
21
D
```

replace(string, replacement): This method searches a string for a specified pattern, and replaces matches with a replacement string.

split(string): This method splits a string into an array of substrings using a regular expression pattern as a delimiter.



map():
The map() method creates a new array populated with the results of calling a provided function on every element in the calling array.



findIndex():
The findIndex() method returns the index of the first element in the array that satisfies the provided testing function. Otherwise, it returns -1.

find():
The find() method returns the value of the first element in the array that satisfies the provided testing function. Otherwise, it returns undefined.

```
1  const numbers = [10, 20, 30, 40, 50];
2
3  const foundNumber = numbers.find(function(num) {
4      return num > 25;
5  });
6
7  console.log(foundNumber);
8
```

```
node /tmp/beTs1pDtyK.js
30
```

filter():
The filter() method creates a new array with all elements that pass the test implemented by the provided function.

```
1  const numbers = [10, 20, 30, 40, 50];
2
3  const filteredNumbers = numbers.filter(function(num) {
4      return num > 25;
5  });
6
7  console.log(filteredNumbers);
8
```

```
node /tmp/elcPE9BJof.js
[ 30, 40, 50 ]
```