Jhanvi Parekh

60009210033

CSE(Data Science)

Experiment No 6

Aim: Implement Web application using Server side and Client Side Scripting language

Theory: -

What is Scripting Language

All scripting languages are programming languages. The scripting language is basically a language where instructions are written for a run time environment. They do not require the compilation step and are rather interpreted. It brings new functions to applications and glue complex system together. A scripting language is a programming language designed for integrating and communicating with other programming languages.

Application of Scripting Languages: Scripting languages are used in many areas:

Scripting languages are used in web applications. It is used in server side as well as client side. Server side scripting languages are: JavaScript, PHP, Perl etc. and client side scripting languages are: JavaScript, AJAX, jQuery etc.

Scripting languages are used in system administration. For example: Shell, Perl, Python scripts etc.

It is used in Games application and Multimedia.

It is used to create plugins and extensions for existing applications.

Difference Between Programming Language, Scripting Language, and mark-up language

All the scripting languages are programming languages but all the programming languages are not scripting languages. C cannot be called a scripting language; it is just a programming language but we can call JavaScript or Php programming or scripting languages.

Also, there is no need to compile scripting languages it only needs to be interpreted. Scripting languages are generally slower than programming languages because compiled programs are first converted into machine code.

On the other hand, markup languages are just used to define the structure of data which doesn’t require any logic or algorithm.

A. Client Side Scripting Language

Client-side programming, such as JavaScript, can be integrated in a client's browser's page. When operating a web application, this script will allow the client's browser to relieve some of the load on your web server. Client-side scripting enables the construction of quicker and more responsive online applications by executing source code on the client's browser rather than the web server.

Client-side scripting refers to a type of online programming that is run on the client side, rather than on the server, by the user's web browser (on the web server).

1. Introduction to Java Script

JavaScript is a lightweight, cross-platform, and interpreted scripting language. It is well-known for the development of web pages; many non-browser environments also use it. JavaScript can be used for Client-side developments as well as Server-side developments. JavaScript contains a standard library of objects, like Array, Date, and Math, and a core set of language elements like operators, control structures, and statements.

Client-side: It supplies objects to control a browser and its Document Object Model (DOM). Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation. Useful libraries for the client-side are AngularJS, ReactJS, VueJS and so many others.

Server-side: It supplies objects relevant to running JavaScript on a server. Like if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is node.js.

JavaScript can be added to your HTML file in two ways:

Internal JS: We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.

External JS: We can write JavaScript code in other file having an extension .js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

Syntax:

```
<script>
   // JavaScript Code
</script>
```

Example of Simple JavaScript Program

```
<! DOCTYPE html>
<html lang="en">
<head>
<title>
        Basic Example to Describe JavaScript
</title>
</head>
```

```
<body>
<! -- JavaScript code can be embedded inside
        head section or body section -->
<script>
        console.log ("Welcome to Java Script ");
</script>
</body>
</html>
```

B. Server side Scripting Language

Server-side scripting is a technique used in web development which involves employing scripts on a web server which produces a response customized for each user's (client's) request to the website.

The alternative is for the web server itself to deliver a static web page. Scripts can be written in any of a number of server-side scripting languages that are available. Server-side scripting is distinguished from client-side scripting where embedded scripts, such as JavaScript, are run client-side in a web browser, but both techniques are often used together.

Server-side scripting is often used to provide a customized interface for the user. These scripts may assemble client characteristics for use in customizing the response based on those characteristics, the user's requirements, access rights, etc. Server-side scripting also enables the website owner to hide the source code that generates the interface, whereas, with client-side scripting, the user has access to all the code received by the client.

It is the program that runs on server dealing with the generation of content of web page.

1) Querying the database

2) Operations over databases

3) Access/Write a file on server.

4) Interact with other servers.

5) Structure web applications.

6) Process user input.

For example, if user input is a text in search box, run a search algorithm on data stored on server and send the results.

Examples:

The Programming languages for server-side programming are:

1) PHP

2) Java(Java Server Pages)

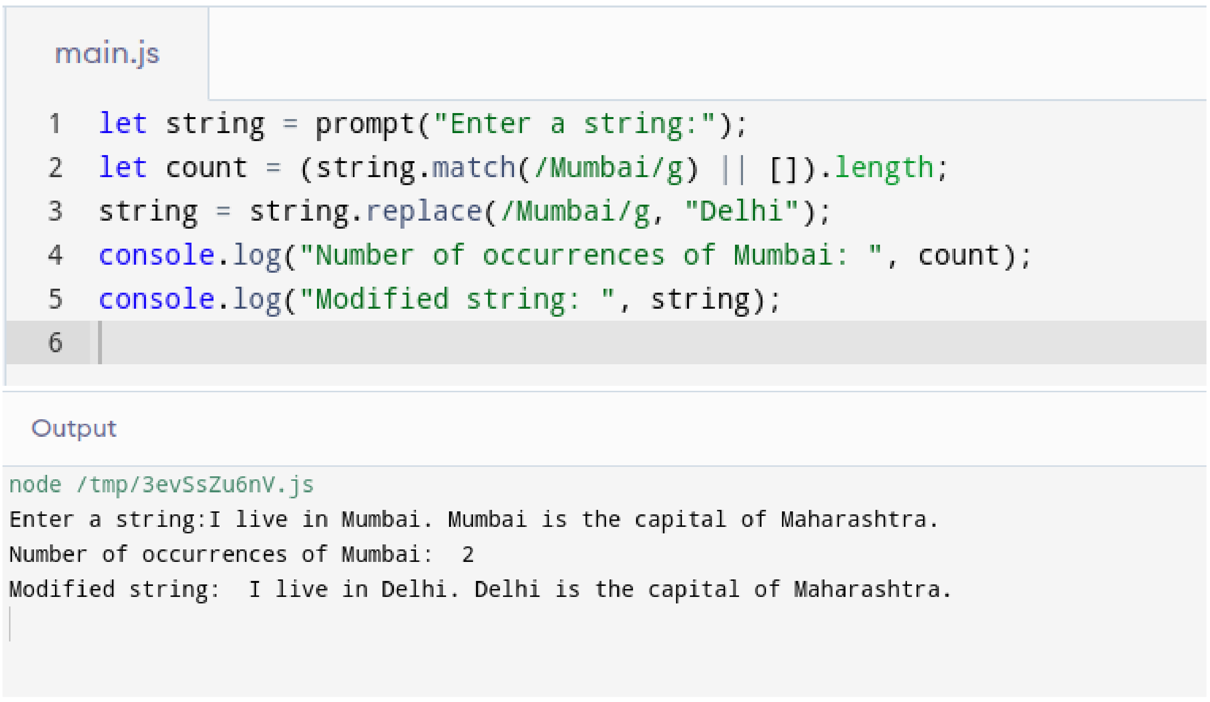
3) Python

4) ASP.net

5) Perl

Lab Assignments to complete in this session

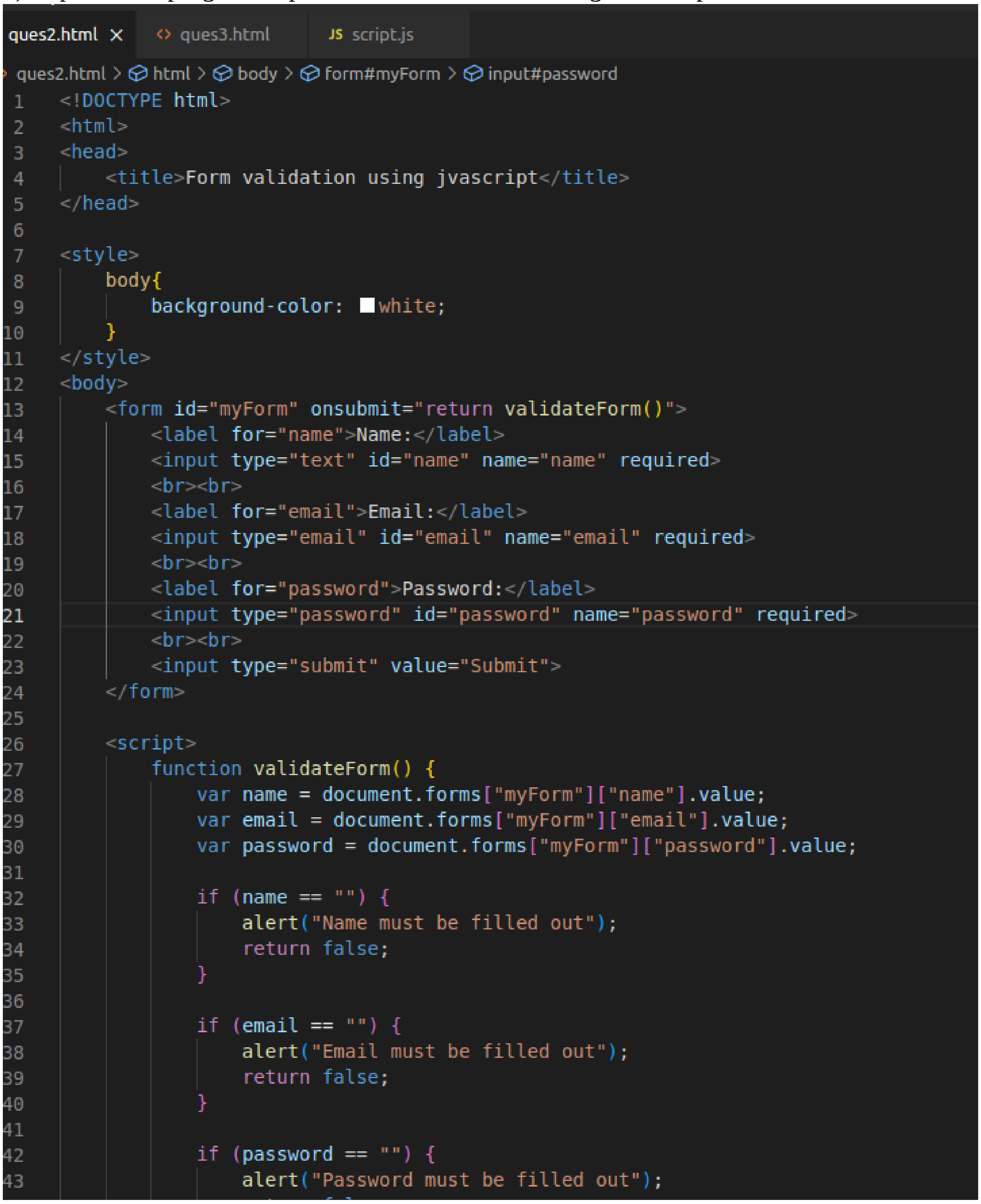
a) Implement a Program to take the string from the user, Find the number of occurrences of Mumbai in the string and replace Mumbai with Delhi. (Python/JavaScript)



```javascript
let string = prompt("Enter a string:");
let count = (string.match(/Mumbai/g) || []).length;
string = string.replace(/Mumbai/g, "Delhi");
console.log("Number of occurrences of Mumbai: ", count);
console.log("Modified string: ", string);
```

```
node /tmp/3evSsZu6nV.js
Enter a string:I live in Mumbai. Mumbai is the capital of Maharashtra.
Number of occurrences of Mumbai:  2
Modified string:  I live in Delhi. Delhi is the capital of Maharashtra.
```

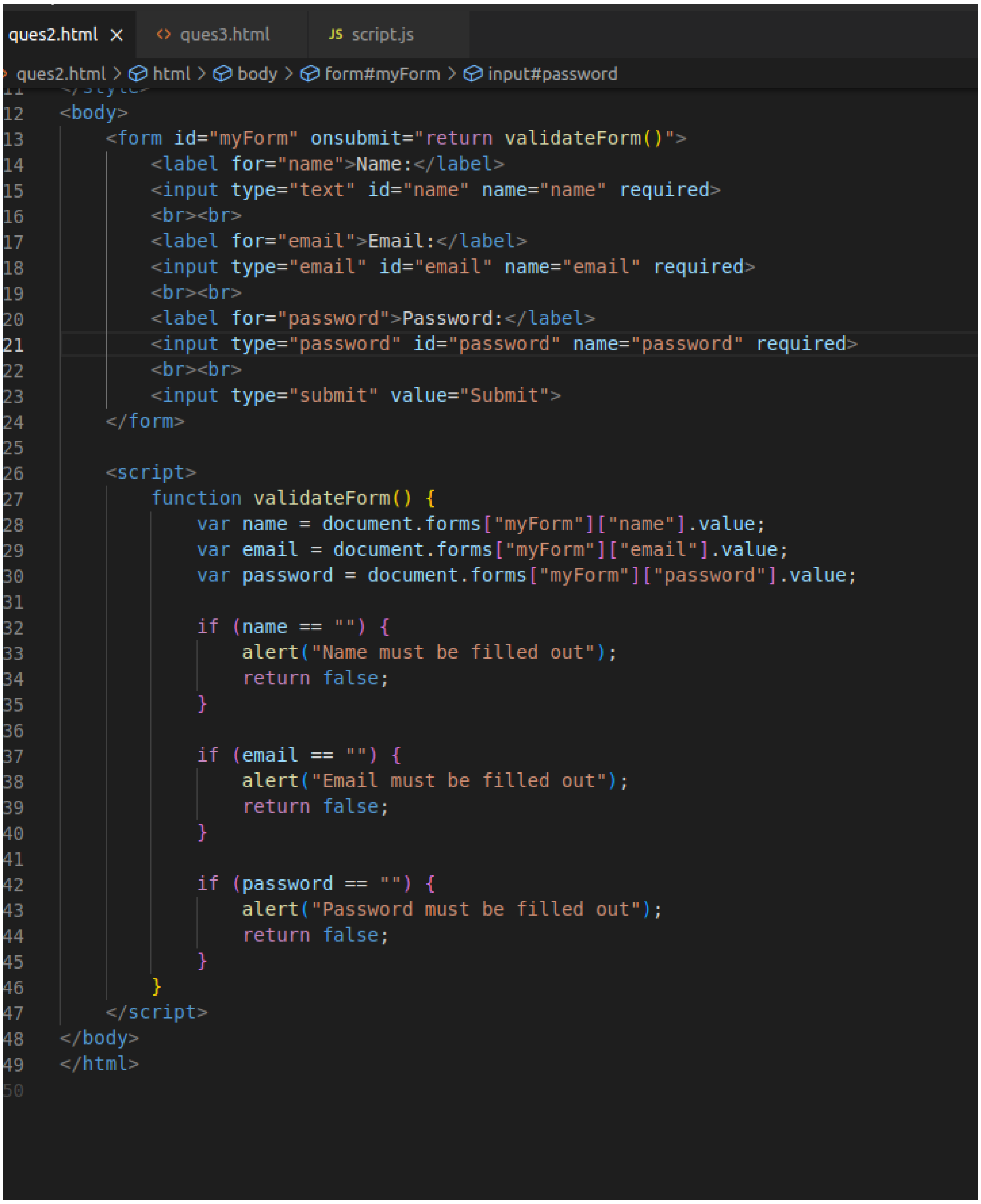
b) Implement a program to perform form validation using JavaScript



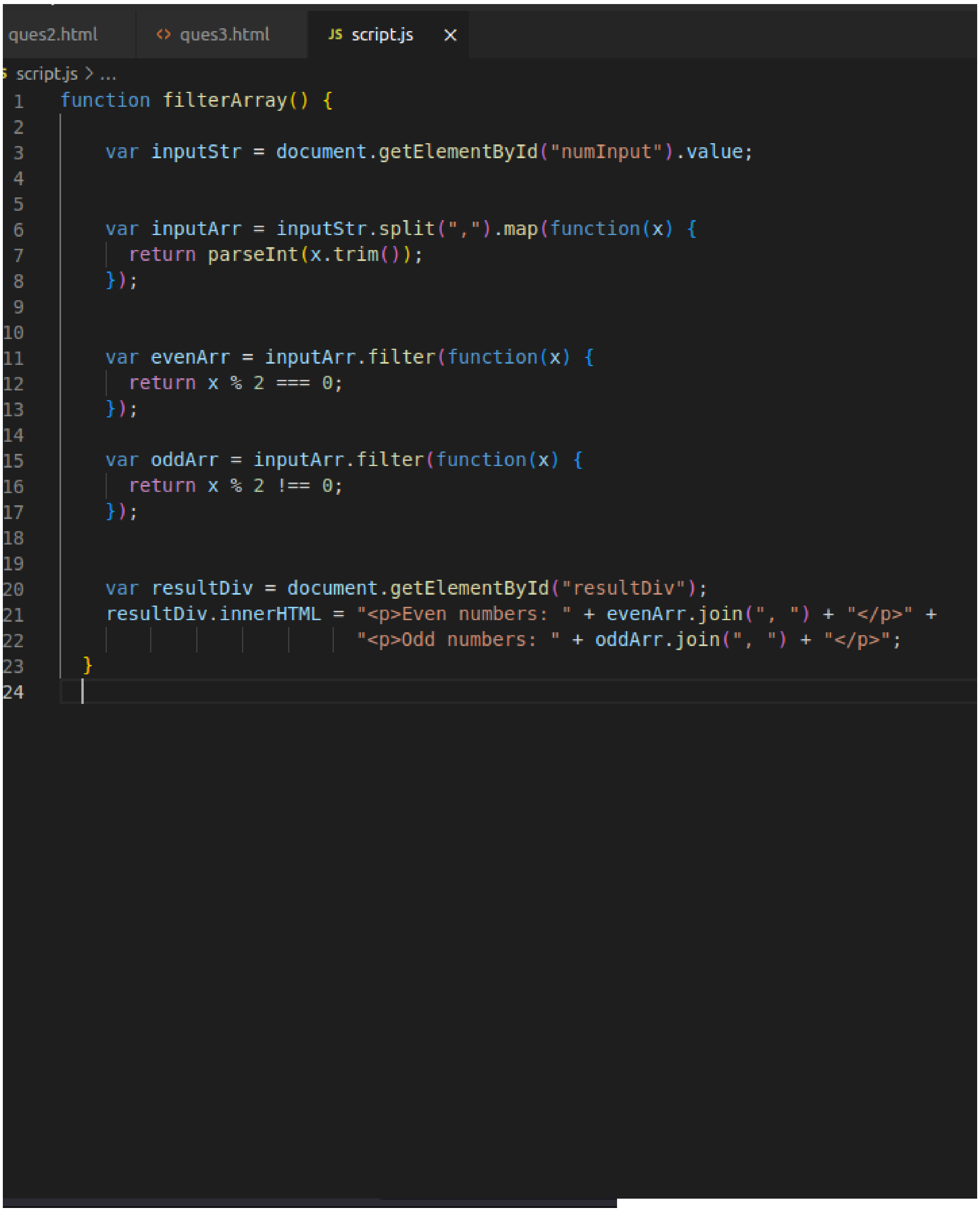
ques2.html × | ques3.html | script.js

ques2.html > html > body > form#myForm > input#password

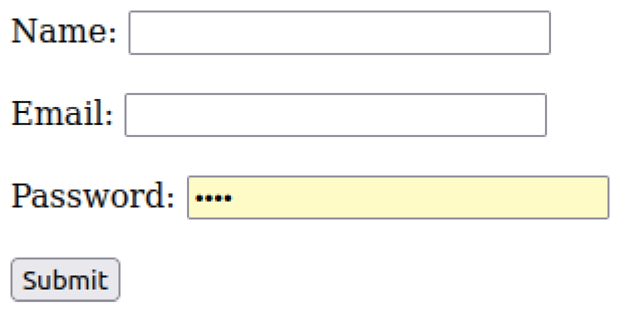
```html
<!DOCTYPE html>
<html>
<head>
    <title>Form validation using jvascript</title>
</head>

<style>
    body{
        background-color: white;
    }
</style>
<body>
    <form id="myForm" onsubmit="return validateForm()">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required>
        <br><br>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <br><br>
        <input type="submit" value="Submit">
    </form>

    <script>
        function validateForm() {
            var name = document.forms["myForm"]["name"].value;
            var email = document.forms["myForm"]["email"].value;
            var password = document.forms["myForm"]["password"].value;

            if (name == "") {
                alert("Name must be filled out");
                return false;
            }

            if (email == "") {
                alert("Email must be filled out");
                return false;
            }

            if (password == "") {
                alert("Password must be filled out");
```



```
<body>
    <form id="myForm" onsubmit="return validateForm()">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required>
        <br><br>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <br><br>
        <input type="submit" value="Submit">
    </form>

    <script>
        function validateForm() {
            var name = document.forms["myForm"]["name"].value;
            var email = document.forms["myForm"]["email"].value;
            var password = document.forms["myForm"]["password"].value;

            if (name == "") {
                alert("Name must be filled out");
                return false;
            }

            if (email == "") {
                alert("Email must be filled out");
                return false;
            }

            if (password == "") {
                alert("Password must be filled out");
                return false;
            }
        }
    </script>
</body>
</html>
```

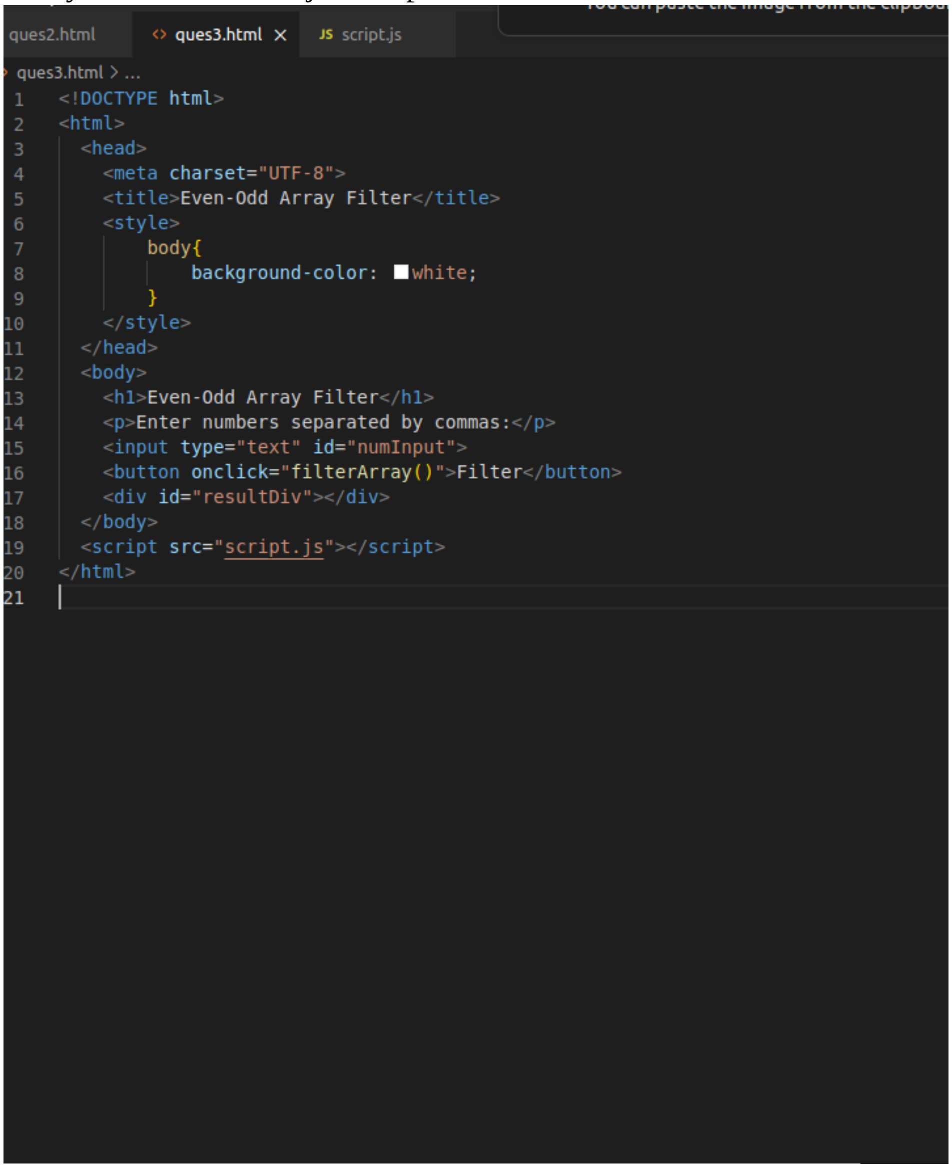


```
function filterArray() {

    var inputStr = document.getElementById("numInput").value;


    var inputArr = inputStr.split(",").map(function(x) {
      return parseInt(x.trim());
    });


    var evenArr = inputArr.filter(function(x) {
      return x % 2 === 0;
    });

    var oddArr = inputArr.filter(function(x) {
      return x % 2 !== 0;
    });


    var resultDiv = document.getElementById("resultDiv");
    resultDiv.innerHTML = "<p>Even numbers: " + evenArr.join(", ") + "</p>" +
                          "<p>Odd numbers: " + oddArr.join(", ") + "</p>";
  }

```

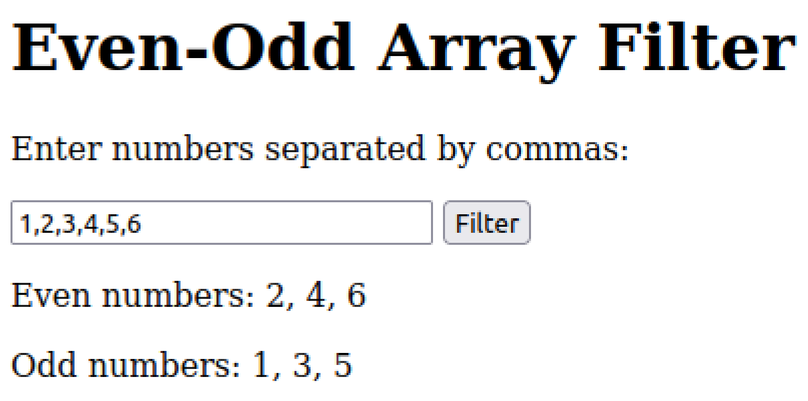


Name:	
Email:	
Password:	••••

c) Write a program to take array and filter the array into even array and odd array using anonymous function in javascript. Create GUI for the same.



```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Even-Odd Array Filter</title>
    <style>
        body{
            background-color: white;
        }
    </style>
  </head>
  <body>
    <h1>Even-Odd Array Filter</h1>
    <p>Enter numbers separated by commas:</p>
    <input type="text" id="numInput">
    <button onclick="filterArray()">Filter</button>
    <div id="resultDiv"></div>
  </body>
  <script src="script.js"></script>
</html>
```



Even-Odd Array Filter

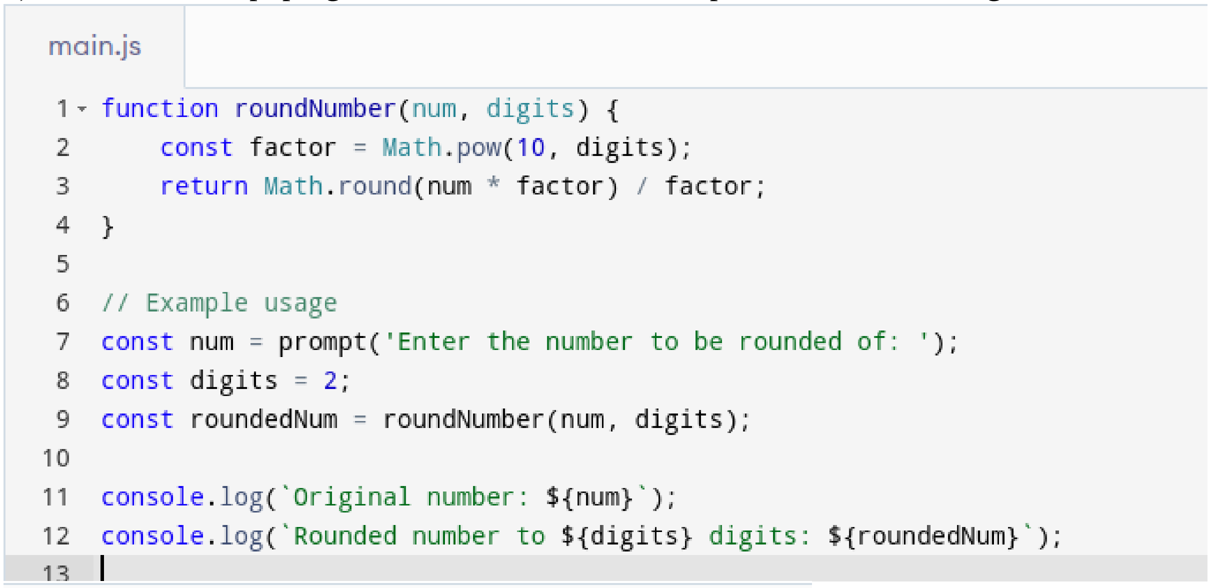
Enter numbers separated by commas:

```
1,2,3,4,5,6  [Filter]
```

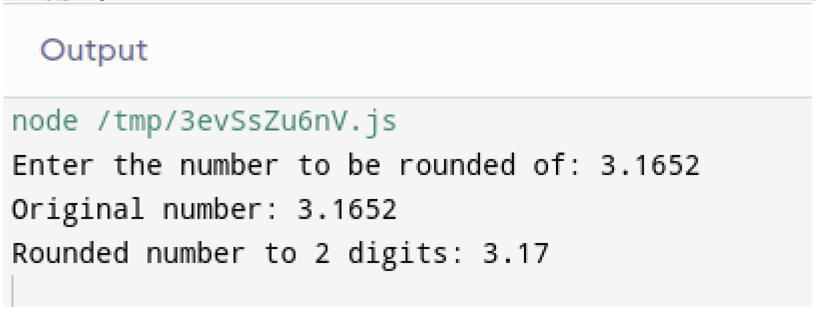
Even numbers: 2, 4, 6

Odd numbers: 1, 3, 5

d) Write a JavaScript program to round a number to a specified number of digits.



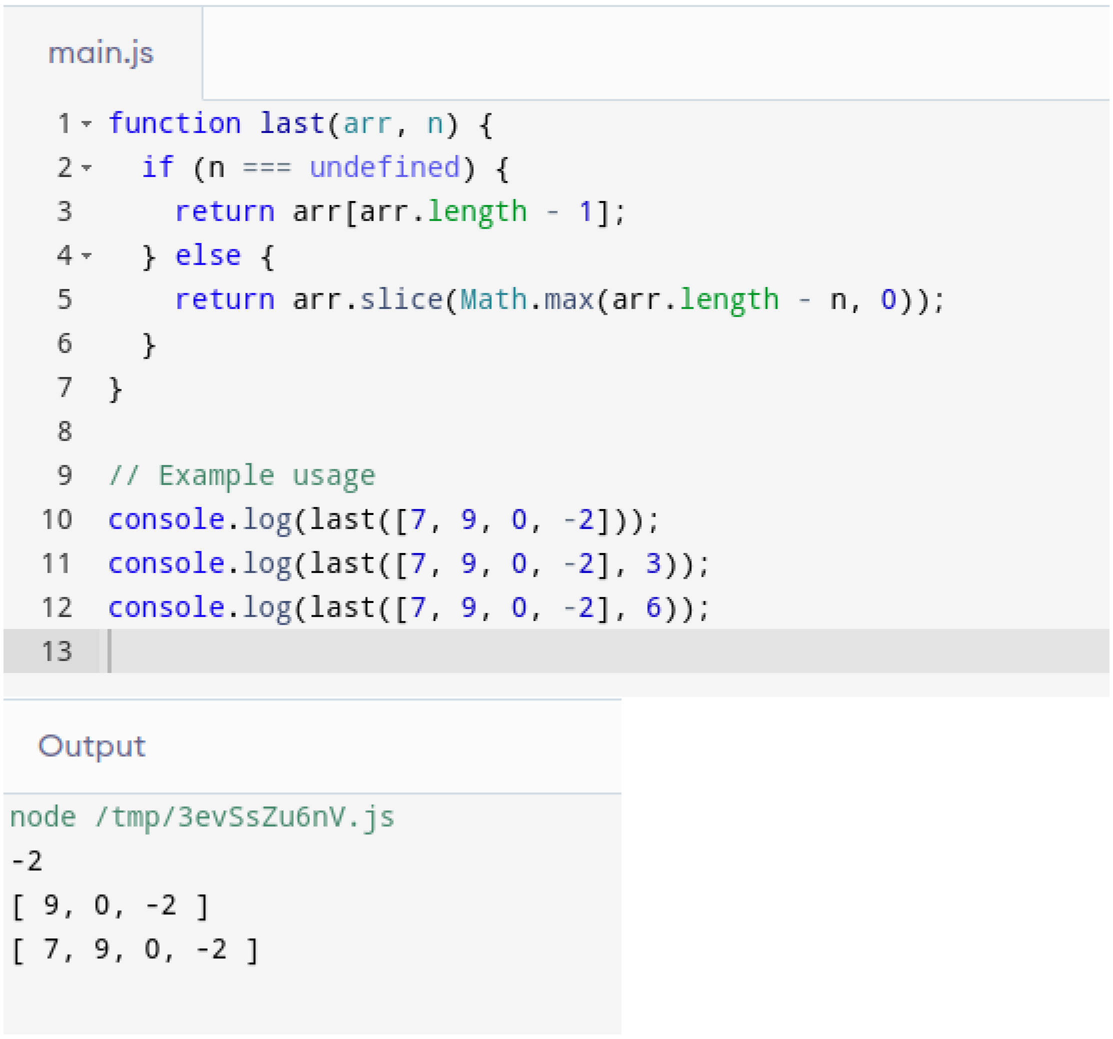
```
function roundNumber(num, digits) {
    const factor = Math.pow(10, digits);
    return Math.round(num * factor) / factor;
}

// Example usage
const num = prompt('Enter the number to be rounded of: ');
const digits = 2;
const roundedNum = roundNumber(num, digits);

console.log(`Original number: ${num}`);
console.log(`Rounded number to ${digits} digits: ${roundedNum}`);
```



```
node /tmp/3evSsZu6nV.js
Enter the number to be rounded of: 3.1652
Original number: 3.1652
Rounded number to 2 digits: 3.17
```

e) Write a JavaScript function to get the last element of an array. Passing a parameter 'n' will return the last 'n' elements of the array.

Test Data :
console.log(last([7, 9, 0, -2]));
console.log(last([7, 9, 0, -2],3));
console.log(last([7, 9, 0, -2],6));
Expected Output :
-2
[9, 0, -2]
[7, 9, 0, -2]


main.js

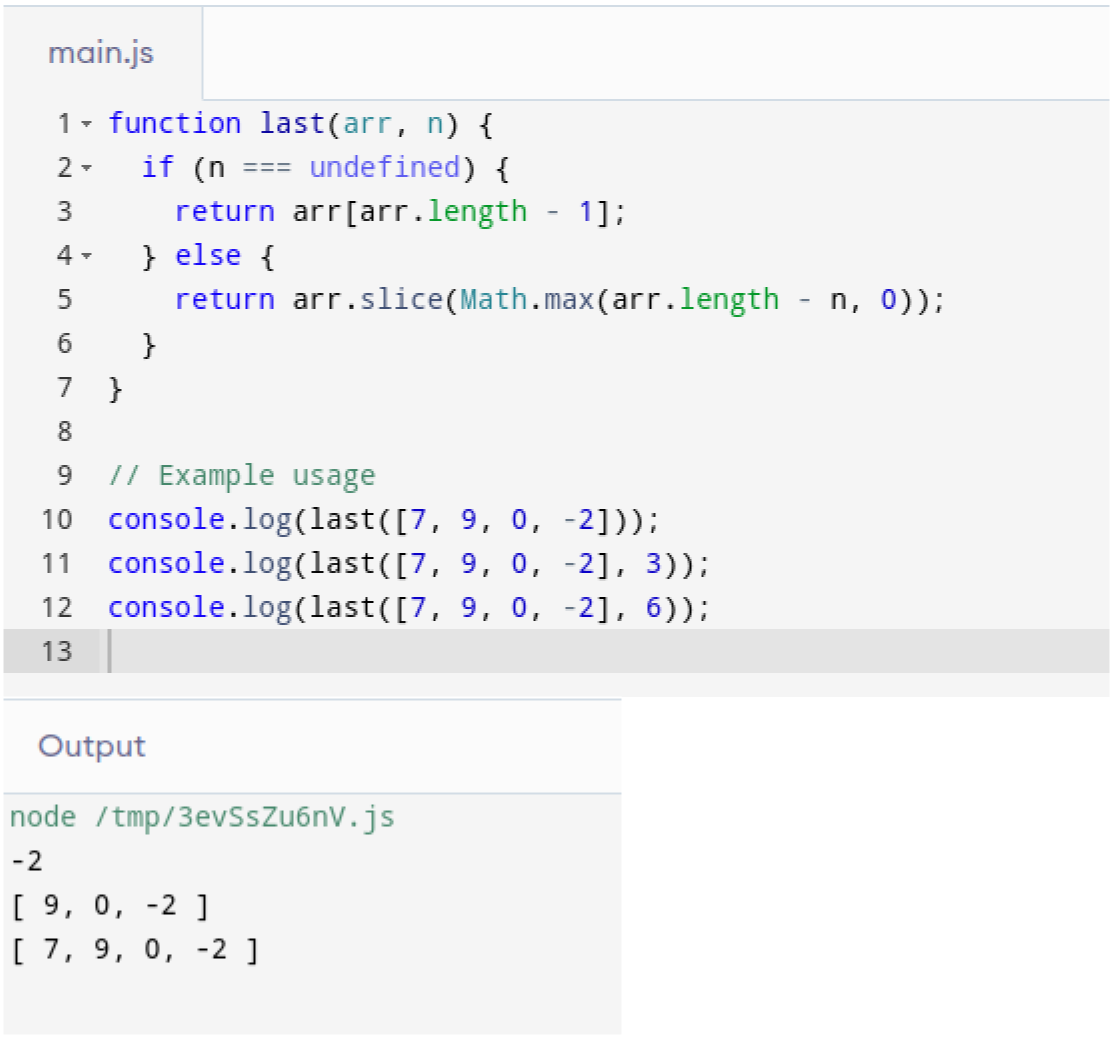
```javascript
function last(arr, n) {
  if (n === undefined) {
    return arr[arr.length - 1];
  } else {
    return arr.slice(Math.max(arr.length - n, 0));
  }
}

// Example usage
console.log(last([7, 9, 0, -2]));
console.log(last([7, 9, 0, -2], 3));
console.log(last([7, 9, 0, -2], 6));
```

Output

```
node /tmp/3evSsZu6nV.js
-2
[ 9, 0, -2 ]
[ 7, 9, 0, -2 ]
```

f) Write a JavaScript function to get the last element of an array. Passing a parameter 'n' will return the last 'n' elements of the array.

Test Data :
console.log(last([7, 9, 0, -2]));
console.log(last([7, 9, 0, -2],3));
console.log(last([7, 9, 0, -2],6));
Expected Output :
-2
[9, 0, -2]
[7, 9, 0, -2]



main.js

```
function last(arr, n) {
  if (n === undefined) {
    return arr[arr.length - 1];
  } else {
    return arr.slice(Math.max(arr.length - n, 0));
  }
}

// Example usage
console.log(last([7, 9, 0, -2]));
console.log(last([7, 9, 0, -2], 3));
console.log(last([7, 9, 0, -2], 6));

```

Output

```
node /tmp/3evSsZu6nV.js
-2
[ 9, 0, -2 ]
[ 7, 9, 0, -2 ]
```