

# Documentation for Reservation Form

## Contents:

- 1.User-Interface
- 2.codes explanation
- 3.Outputs
- 4.Github Link

UI looks like:

**Reservation Form**

Date:

Time:

Name:

Email:

Phone number:

Number of guests:

## Explanation of HTML code:

1. `<!DOCTYPE html>`: This declares the document type as HTML5.
2. `<html lang="en">`: This is the root element of the HTML document. The `lang` attribute specifies the language of the document.
3. `<head>`: This element contains metadata about the document, including the title, character set, and other information that the browser may need to interpret the document.
4. `<meta charset="UTF-8">`: This specifies the character encoding of the document as UTF-8, which is a common character encoding that can represent many languages.
5. `<meta http-equiv="X-UA-Compatible" content="IE=edge">`: This sets the compatibility mode for Internet Explorer to the latest version.
6. `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: This sets the viewport properties for the document. The `width=device-width` attribute ensures that the document is displayed at the correct width on mobile devices, while the `initial-scale=1.0` attribute sets the initial zoom level to 100%.
7. `<title>Reservation</title>`: This sets the title of the document to "Reservation".
8. `<link rel="stylesheet" href="mujahid.css">`: This links to an external CSS file called "mujahid.css" that contains styles for the document.
9. The `<body>` tag indicates the start of the document's body content.

10. The `<form>` tag creates a form element that contains the input fields for the reservation form. The `id` attribute "reservation-form" is assigned to the form, which can be used later to target the form element in CSS or JavaScript.
11. The `<h2>` tag creates a heading that displays "Reservation Form" at the top of the form.
12. The `<label>` tags create labels for each input field. The "for" attribute is assigned to each label and links it to the corresponding input field using the "id" attribute. This creates a semantic association between the label and input field that can be used by screen readers or other assistive technologies to improve accessibility.
13. The `<input>` tags create the input fields for the form. Each input field has a "type" attribute that specifies the type of input field (e.g., date, time, text, email, tel, number). The "id" attribute is used to link the input field with its corresponding label. The "name" attribute is used to identify the input field when the form is submitted to the server. The "placeholder" attribute provides a hint to the user about what to enter in the input field. The "required" attribute indicates that the input field must be filled out before the form can be submitted. The "min" and "max" attributes specify the minimum and maximum values for the number of guests input field.
14. The `<input>` tag with `type="submit"` creates the submit button for the form. When the user clicks this button, the form data will be submitted to the server for processing.
15. The `<script>` tag includes a JavaScript file named "mujahid.js". This file could contain scripts that manipulate the form data or perform other actions on the page.

## Explanation of CSS code:

1. The first section applies a background image to the entire body of the webpage and sets the background-repeat to no-repeat, positions the image in the center, and adjusts its size to cover the entire background.
2. The "form" selector sets the maximum width to 500px, centers the form horizontally with the "margin" property, adds padding to the form for spacing and readability, sets a background color to the form, adds a box-shadow effect around the form, and applies a linear gradient background image to the form.
3. The "h2" selector aligns the text to the center, sets the font size to 3rem, adds a margin-bottom of 40px, and sets the text color to a specific shade of green.
4. The "label" selector displays the labels as blocks, sets the font size to 1.5rem, sets the font weight to 600, adds a margin-bottom of 10px, and sets the text color to black.
5. The "input" selector applies styles to various input types such as date, time, number, text, email, and telephone. It displays them as blocks, sets the width to 100%, adds padding for spacing, adds a margin-bottom of 30px for spacing, sets a border-radius of 5px for rounded corners, removes any borders, sets the background color to a specific shade of grey, sets the text color to black, sets the font size to 1.2rem, sets the font weight to 400, and adds a box-shadow effect around the input fields.
6. The "input: focus" selector applies styles to input fields when they are in focus. It removes the outline around the input fields and adds a box-shadow effect with a specific color.
7. The "input[type='submit']" selector applies styles to the submit button of the form. It sets the display property to block, adds a margin-top of 60px, sets

the background color to a specific shade of green, sets the text color to white, removes any borders, sets the border-radius to 10px for rounded corners, adds padding for spacing, sets the font size to 1.3rem, sets the font weight to 600, sets the cursor to pointer to show that the button is clickable, and adds a transition effect to the background color.

8. The "input[type='submit']: hover" selector applies styles to the submit button when it is hovered over with the mouse. It changes the background color to a darker shade of green to indicate interactivity.

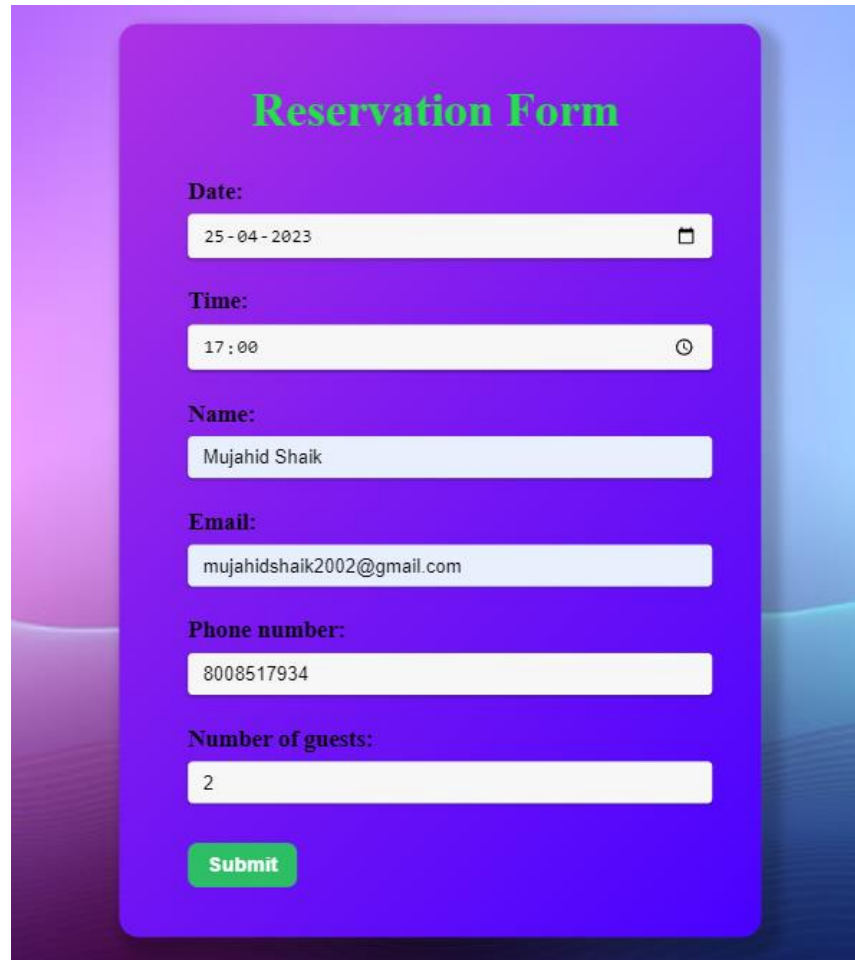
## **Explanation of JS code:**

1. The first line of code uses the `document.getElementById()` method to select the reservation form element on the page and assign it to the `reservationForm` constant.
2. The `reservations` array is initialized as an empty array.
3. An event listener is added to the `reservationForm` element to listen for a "submit" event. When the form is submitted, the event listener's callback function is executed.
4. The first line of the callback function uses the `e.preventDefault()` method to prevent the default behavior of the form submission, which is to refresh the page.
5. The next few lines of code retrieve the values of the form fields using the `document.getElementById().value` method and assign them to constants with meaningful names.

6. The next few lines of code validate the name, email, and phone fields. If any of these fields are invalid, an alert message is displayed and the function returns early without executing any further code.
7. The next few lines of code create a reservation object containing the reservation details and add it to the reservations array.
8. If the reservations array contains 20 or more reservations, an alert message is displayed, and the function returns early.
9. If the reservation is successful, the form is reset using the `reservationForm.reset()` method, and a success message is displayed using the `alert()` method.
10. The reservation details are also logged to the console using the `console.log()` method.

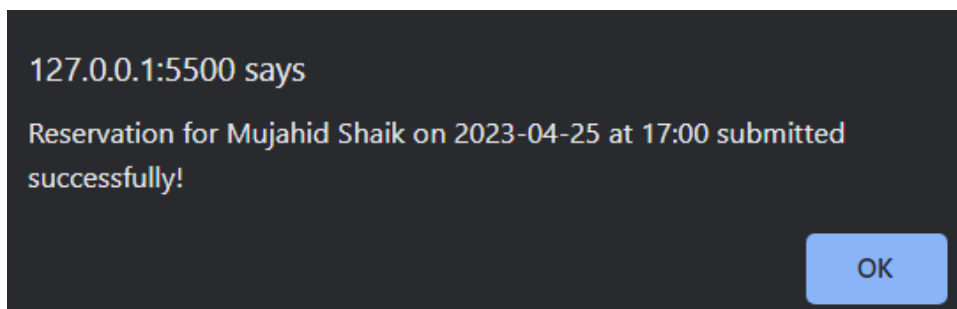
## OUTPUTS

### **1.Filled Data:**



A screenshot of a web application's reservation form. The form is titled "Reservation Form" in a large, bold, green font. It contains several input fields with labels: "Date:" with a date picker showing "25 - 04 - 2023", "Time:" with a time picker showing "17 : 00", "Name:" with a text input containing "Mujahid Shaik", "Email:" with a text input containing "mujahidshaik2002@gmail.com", "Phone number:" with a text input containing "8008517934", and "Number of guests:" with a text input containing "2". At the bottom of the form is a green "Submit" button. The form is set against a dark blue background with a subtle wave pattern.

**2.After submit button:**



**3.In Console Output:**

```
▼ Array(1) ⓘ
  ▼ 0:
    date: "2023-04-25"
    email: "mujahidshaik2002@gmail.com"
    guests: "2"
    name: "Mujahid Shaik"
    phone: "8008517934"
    time: "17:00"
    ► [[Prototype]]: Object
  length: 1
  ► [[Prototype]]: Array(0)
```

#### 4.In Console for multiple inputs :

```
mujahid.js:47
▼ Array(4) ⓘ
  ► 0: {date: '2023-04-25', time: '17:00', name: 'Mujahid Shaik', email: 'mujahidshaik2002@gmail.com', phone: '8008517934', ...}
  ► 1: {date: '2023-04-25', time: '19:10', name: 'Sowjanya', email: 'sowji@gmail.com', phone: '8008517934', ...}
  ► 2: {date: '2023-04-26', time: '22:15', name: 'PHN', email: 'phn@gmail.com', phone: '0123456789', ...}
  ► 3: {date: '2023-04-28', time: '20:10', name: 'web dev', email: 'intern@gmail.com', phone: '8008517934', ...}
  length: 4
```

#### 5.If User enters invalid inputs the errors appeared in the form of “alert”:

##### Multiple errors are:

```
127.0.0.1:5500 says
Name should only contain letters and spaces not like Muja-hid Sha1k
OK
```



127.0.0.1:5500 says

Please enter a valid 10-digit phone number not like M80085179

OK

**Number of guests:**

20



Value must be less than or equal to 10.

**Submit**

4. GitHub Link: Click [here](#)

**Thank You**