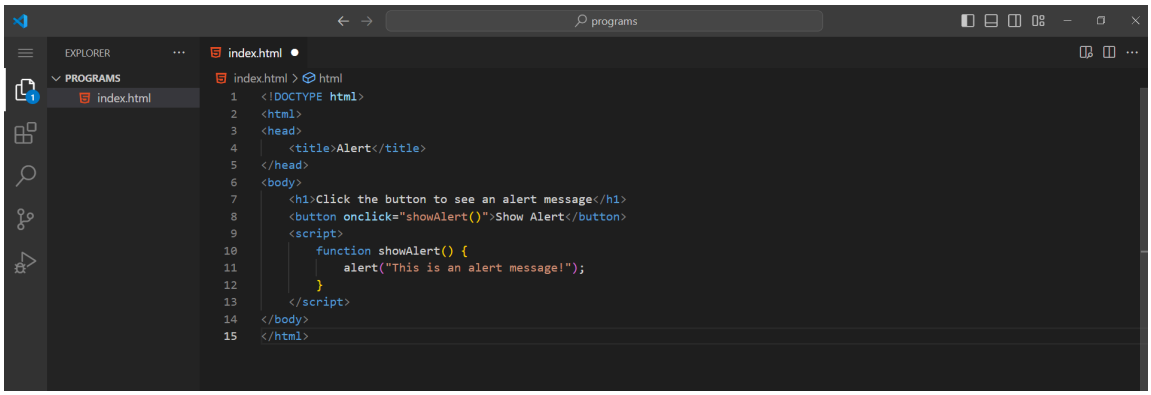
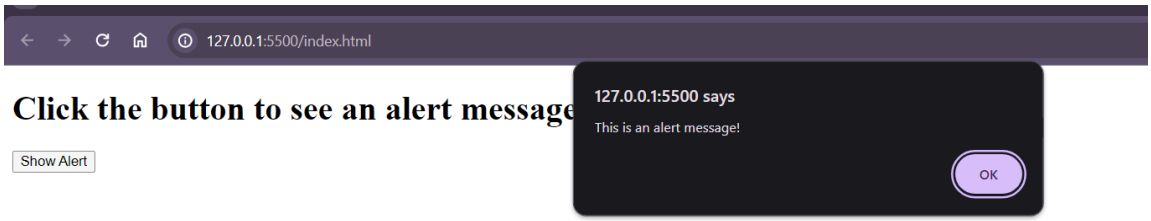
Advance JavaScript for Front-End Introduction and Code Quality

1. **Write a program to Show an alert.**





2. **What will be the result for these expressions?**

1)5 > 4:
This is a straightforward numerical comparison.
Result: true because 5 is greater than 4.

2)"apple" > "pineapple":
This compares strings lexicographically (dictionary order).
Result: false because in lexicographical order, "apple" comes before "pineapple" (since "a" is less than "p").

3)"2" > "12":
This compares strings lexicographically.
Result: true because in lexicographical order, "2" is greater than "1" (the first character comparison determines the result).

4)undefined == null:
This checks for equality with type conversion.
Result: true because undefined and null are considered equal with ==.

5)undefined === null:
This checks for strict equality (without type conversion).
Result: false because undefined and null are of different types.

6)null == "\n0\n":
This checks for equality with type conversion.
Result: false because null only equals undefined with ==, and does not equal any string.

7)null === +"\n0\n":
This checks for strict equality.
+"\n0\n" converts the string to a number.
+"\n0\n" results in 0 because the string is parsed as a number.
Result: false because null is not strictly equal to 0 (different types).

3. **Will alert be shown? if ("0") { alert('Hello'); }**
   - Yes, an alert will be shown. The expression inside the if statement is "0", which is a non-empty string.
   - In JavaScript, any non-empty string is considered truthy.
   - Since "0" is a non-empty string, it evaluates to true in a boolean context.
   - Therefore, the code inside the if block will execute, resulting in the alert being shown.
   - When this code runs, the alert with the message Hello will be displayed.

4. **What is the code below going to output? alert(null || 2 || undefined);**
   - The || (logical OR) operator in JavaScript evaluates operands from left to right and returns the first truthy value it encounters. If all values are falsy, it returns the last value. So,
   - null is a falsy value.
   - 2 is a truthy value.
   - undefined is a falsy value.
   - Since null is falsy, the || operator moves to the next operand, 2 which is truthy. Therefore, it stops there and returns 2.
   - So, the alert will display 2.

5. **The following function returns true if the parameter age is greater than 18. Otherwise it asks for a confirmation and returns its result: function checkAge(age) { else { } } if (age> 18) { return true; } // ...return confirm (‘did parents allow you?');**

6. **Replace Function Expressions with arrow functions in the code below: Function ask(question, yes, no) { if (confirm(question))yes(); else no(); } ask("Do you agree?", function() { alert("You agreed."); }, function() { alert("You canceled the execution."); } } Data Types and Objects**

7. **Write the code, one line for each action: a) Create an empty object user. b) Add the property name with the value John. c) Add the property surname with the value Smith. d) Change the value of the name to Pete. e) Remove the property name from the object.**

8. **Is array copied? let fruits = ["Apples", "Pear", "Orange"]; // push a new value into the "copy" let shoppingCart = fruits; shoppingCart.push("Banana"); // what's in fruits? alert(fruits.length); // ?**

9. **Map to names let john = { name: "John", age: 25 }; let pete = { name: "Pete", age: 30 }; let mary = { name: "Mary", age: 28 }; let users = [john, pete, mary]; let names = /* ... your code */ alert(names); // John, Pete, Mary**

10. **Map to objects let john = { name: "John", surname: "Smith", id: 1 }; let pete = { name: "Pete", surname: "Hunt", id: 2 }; let mary = { name: "Mary", surname: "Key", id: 3 }; let users = [john, pete, mary]; let usersMapped = /* ... your code ... */ /* usersMapped = [{ fullName: "John Smith", id: 1 }, { fullName: "Pete Hunt", id: 2 }, { fullName: "Mary Key", id: 3 }] */ alert(usersMapped[0].id) // 1 alert(usersMapped[0].fullName) // John Smith**

11. **Sum the properties There is a salaries object with an arbitrary number of salaries. Write the function sumSalaries(salaries) that returns the sum of all salaries using Object.values and the for..of loop.If salaries is empty, then the result must be 0. let salaries = { "John": 100, "Pete": 300, "Mary": 250 }; alert(sumSalaries(salaries)); // 650**

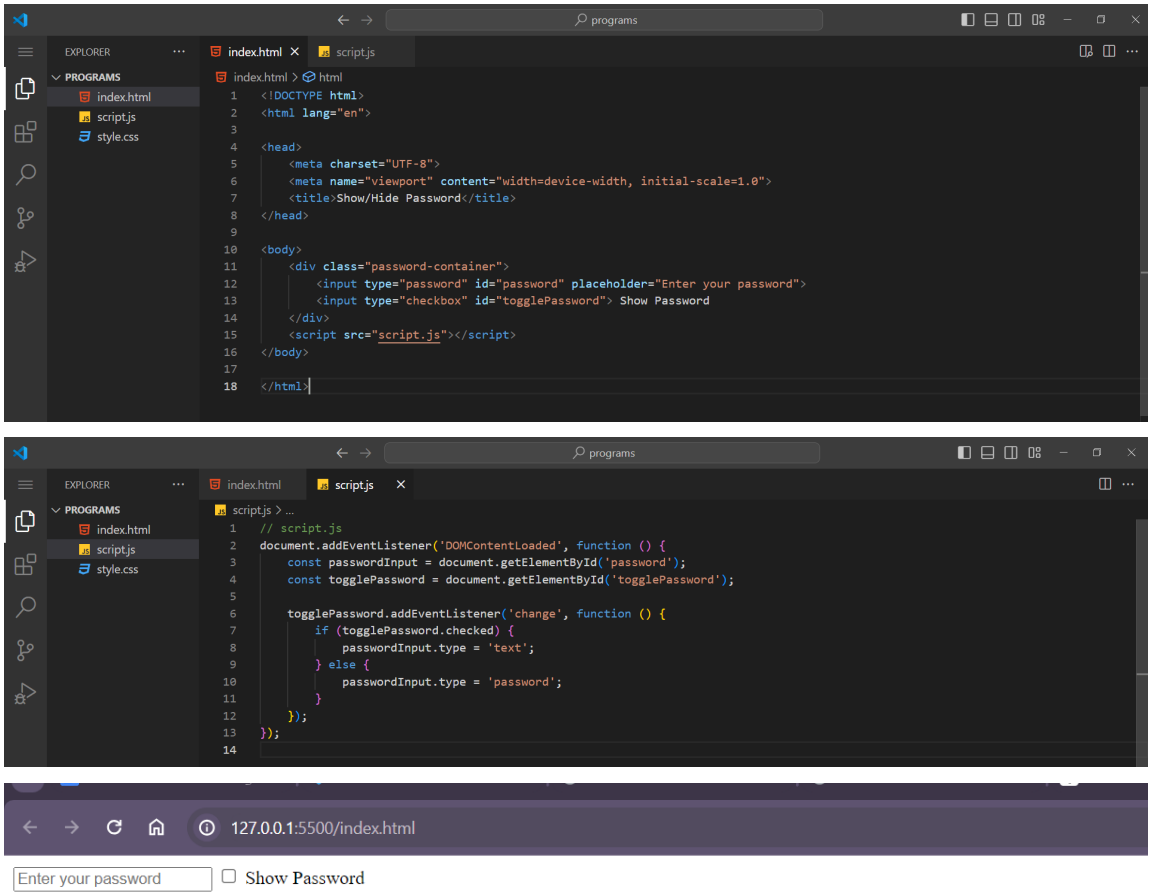
12. **Destructuring assignment We have an object: Write the Destructuring assignment that reads: a) Name property into the variable name. b) Year's property into the variable age. c) isAdmin property into the variable isAdmin (false, if no such property) d) let user = { name: "John", years: 30};**

13. **Turn the object into JSON and back Turn the user into JSON and then read it back into another variable. user = { name: "John Smith", age: 35};**

Document, Event and Controls

Document, Event and Controls

14. Create a program to hide/show the password



```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Show/Hide Password</title>
</head>

<body>
    <div class="password-container">
        <input type="password" id="password" placeholder="Enter your password">
        <input type="checkbox" id="togglePassword"> Show Password
    </div>
    <script src="script.js"></script>
</body>

</html>
```



```
// script.js
document.addEventListener('DOMContentLoaded', function () {
    const passwordInput = document.getElementById('password');
    const togglePassword = document.getElementById('togglePassword');

    togglePassword.addEventListener('change', function () {
        if (togglePassword.checked) {
            passwordInput.type = 'text';
        } else {
            passwordInput.type = 'password';
        }
    });
});
```


New Request

15. What is JSON

- JSON stands for JavaScript Object Notation.
- It is easy to understand.
- It is a lightweight format for storing and transporting the data.
- It is often used when data is sent from server to web page.

16. What is promises

- A JavaScript Promise object can be pending,fulfilled and rejected.
- The Promise object supports two properties: state and result.
- While a Promise object is "pending" (working), the result is undefined.
- When a Promise object is "fulfilled", the result is a value.
- When a Promise object is "rejected", the result is an error object.

17. **Write a program of promises and handle that promises also**

18. **Use fetch method for calling an api https://fakestoreapi.com/products**

19. **Display all the products from the api in your HTML page.**