3/11/2023

|  |
| --- |
| **Advance JavaScript** |
| **MODULE: 1 (Introduction and Code Quality)** |
|  |

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

Ans. alert("This is my custom message!");

This will show a pop-up alert dialog box with the message "This is my custom message!".

Note that the **alert** function is a part of the **window** object in a web browser. Therefore, you can call it directly without the need for any additional setup or imports.

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

Ans

5 > 4 => true This expression compares if the value 5 is greater than the value 4. Since 5 is indeed greater than 4, the expression evaluates to true.

1. "apple" > "pineapple" => false This expression compares the two string values "apple" and "pineapple" using lexicographical order. In lexicographical order, "apple" comes before "pineapple", so "apple" is less than "pineapple". Therefore, the expression evaluates to false.
2. "2" > "12" => true This expression compares the two string values "2" and "12" using lexicographical order. In lexicographical order, "2" comes after "1", so "2" is greater than "12". Therefore, the expression evaluates to true.
3. undefined == null => true The == operator performs type coercion, which means that it converts the operands to a common type before comparing them. In this case, undefined and null are both considered "falsy" values in JavaScript, so when they are compared with the == operator, they are considered equal. Therefore, the expression evaluates to true.
4. undefined === null => false The === operator is a strict equality operator, which means that it does not perform type coercion. In this case, undefined and null are two distinct types in JavaScript, so they are not equal when compared with the === operator. Therefore, the expression evaluates to false.
5. null == "\n0\n" => false The == operator performs type coercion, so it converts the operands to a common type before comparing them. In this case, null is considered "falsy", while "\n0\n" is considered "truthy" because it is a non-empty string. Therefore, when they are compared with the == operator, they are not considered equal. Therefore, the expression evaluates to false.
6. null === +"\n0\n" => false The === operator is a strict equality operator, so it does not perform type coercion. In this case, null and +"\n0\n" are two distinct types in JavaScript, so they are not equal when compared with the **7**.===operator. The + operator converts the string "\n0\n" to a number, which is 0. Therefore, the expression evaluates to false.

**3).Will alert be shown? if ("0") { alert( 'Hello'); }**

**Ans.** if ("0") {

alert('Hello');

}

This is because the string "0" is a truthy value in JavaScript, which means that it is considered "true" in a boolean context. Therefore, the code inside the if statement block will be executed, and the **alert** function will be called, displaying the message "Hello" in an alert dialog box.

It's worth noting that not all non-empty strings are truthy in JavaScript. In fact, the only falsy string values are an empty string ("") and the string "false". All other non-empty strings, including "0", are truthy.

**4)What is the code below going to output? alert( null || 2 || undefined );**

**ans.**

It will output the value 2.

This is because the || operator in JavaScript returns the first truthy value it encounters when evaluating the expressions from left to right. In this case, null is a falsy value, so the next value in line, 2, is returned because it is truthy. The final value, undefined, is never reached because 2 is already truthy and therefore satisfies the condition. Therefore, the alert dialog will display the value 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)

{

if (age> 18) { return true; }

else {

// ...return confirm (‘did parents allow you?'}

**Ans.**

The function takes a single parameter age and checks whether it is greater than 18 using an if statement. If the condition is true, the function returns true. If the condition is false, the function executes the code inside the else block.

In the else block, the function displays a confirmation dialog box using the confirm() function with the message "did parents allow you?". The confirm() function returns a boolean value representing the user's response to the dialog box. If the user clicks "OK", the function returns true. If the user clicks "Cancel", the function returns false.

Here's an example usage of the checkAge() function:

In this example, if the age variable is greater than 18, the function will return true and the message "Access granted." will be printed to the console. If the age variable is 18 or less, the function will display a confirmation dialog asking for permission and return the user's response, either true or false. If the user clicks "OK", the message "Access granted." will be printed to the console. If the user clicks "Cancel", the message "Access denied." will be printed instead.

**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."); }

**ans.**

Here's the updated code with arrow functions replacing the original function expressions:

In the updated code, the ask function is now defined using an arrow function instead of a regular function expression. Similarly, the yes and no callbacks are now defined as arrow functions instead of regular function expressions.

Note that the code inside the ask function remains the same, as it doesn't depend on the type of function expression used. The only difference is in the syntax used to define the functions.

Thank you