**Javascript Programming**

Q1- **Practice comparing different values**

In the last two challenges, we learned about the equality operator (==) and the strict equality operator (===). Let's do a quick review and practice using these operators some more.

If the values being compared are not of the same type, the equality operator will perform a type conversion, and then evaluate the values. However, the strict equality operator will compare both the data type and value as-is, without converting one type to the other.

**Examples**

3 == '3' returns true because JavaScript performs type conversion from string to number. 3 === '3' returns false because the types are different and type conversion is not performed.

**Note:** In JavaScript, you can determine the type of a variable or a value with the typeof operator, as follows:

typeof 3

typeof '3'

typeof 3 returns the string number, and typeof '3' returns the string string.

The compareEquality function in the editor compares two values using the equality operator. Modify the function so that it returns the string Equal only when the values are strictly equal.

Hint-

* compareEquality(10, "10") should return the string Not Equal
* compareEquality("20", 20) should return the string Not Equal
* You should use the === operator

**Code:**

**function compareEquality(a,b) {**

**if (a === b) {**

**return "Equal";**

**}**

**return "Not equal";**

**}**

**compareEquality(10,"10")**

Q2- **Comparison with the Inequality Operator**

The inequality operator (!=) is the opposite of the equality operator. It means not equal and returns false where equality would return true and *vice versa*. Like the equality operator, the inequality operator will convert data types of values while comparing.

**Examples**

1 != 2

1 != "1"

1 != '1'

1 != true

0 != false

In order, these expressions would evaluate to true, false, false, false, and false.

Add the inequality operator != in the if statement so that the function will return the string Not Equal when val is not equivalent to 99

Hint-

* testNotEqual(99) should return the string Equal
* testNotEqual("99") should return the string Equal
* testNotEqual(12) should return the string Not Equal
* testNotEqual("12") should return the string Not Equal
* testNotEqual("bob") should return the string Not Equal
* You should use the != operator

**Code:**

**function testNotEqual(val) {**

**if (val != 99) {**

**return "Not Equal";**

**}**

**return "Equal";**

**}**

**testNotEqual(10);**

Q3- **Comparison with the Strict Inequality Operator**

The strict inequality operator (!==) is the logical opposite of the strict equality operator. It means "Strictly Not Equal" and returns false where strict equality would return true and *vice versa*. The strict inequality operator will not convert data types.

**Examples**

3 !== 3

3 !== '3'

4 !== 3

In order, these expressions would evaluate to false, true, and true.

Add the strict inequality operator to the if statement so the function will return the string Not Equal when val is not strictly equal to 17

Hint-

* testStrictNotEqual(17) should return the string Equal
* testStrictNotEqual("17") should return the string Not Equal
* testStrictNotEqual(12) should return the string Not Equal
* testStrictNotEqual("bob") should return the string Not Equal
* You should use the !== operator

**Code:**

**function testStrictNotEqual(val) {**

**if (val !== 99) {**

**return "Not Equal";**

**}**

**return "Equal";**

**}**

**testStrictNotEqual(10);**

Q4- **Comparison with the Greater Than Operator**

The greater than operator (>) compares the values of two numbers. If the number to the left is greater than the number to the right, it returns true. Otherwise, it returns false.

Like the equality operator, the greater than operator will convert data types of values while comparing.

**Examples**

5 > 3

7 > '3'

2 > 3

'1' > 9

In order, these expressions would evaluate to true, true, false, and false.

Add the greater than operator to the indicated lines so that the return statements make sense.

Hint-

* testGreaterThan(0) should return the string 10 or Under
* testGreaterThan(10) should return the string 10 or Under
* testGreaterThan(11) should return the string Over 10
* testGreaterThan(99) should return the string Over 10
* testGreaterThan(100) should return the string Over 10
* testGreaterThan(101) should return the string Over 100
* testGreaterThan(150) should return the string Over 100
* You should use the > operator at least twice

**Code:**

**function testGreaterThan(val) {**

**if (val > 100) {**

**return "Over 100";**

**}**

**if (val > 10) {**

**return "Over 10";**

**}**

**return "10 or Under";**

**}**

**testGreaterThan(10);**

Q5- **Comparison with the Greater Than Or Equal To Operator**

The greater than or equal to operator (>=) compares the values of two numbers. If the number to the left is greater than or equal to the number to the right, it returns true. Otherwise, it returns false.

Like the equality operator, the greater than or equal to operator will convert data types while comparing.

**Examples**

6 >= 6

7 >= '3'

2 >= 3

'7' >= 9

In order, these expressions would evaluate to true, true, false, and false.

Add the greater than or equal to operator to the indicated lines so that the return statements make sense.

Hint-

* testGreaterOrEqual(0) should return the string Less than 10
* testGreaterOrEqual(9) should return the string Less than 10
* testGreaterOrEqual(10) should return the string 10 or Over
* testGreaterOrEqual(11) should return the string 10 or Over
* testGreaterOrEqual(19) should return the string 10 or Over
* testGreaterOrEqual(100) should return the string 20 or Over
* testGreaterOrEqual(21) should return the string 20 or Over
* You should use the >= operator at least twice

**Code:**

**function testGreaterOrEqual(val) {**

**if (val >= 20) {**

**return "20 or Over";**

**}**

**if (val >= 10) {**

**return "10 or Over";**

**}**

**return "9 or Under";**

**}**

**testGreaterOrEqual(10);**

Q6-**Comparison with the Less Than Or Equal To Operator**

The less than or equal to operator (<=) compares the values of two numbers. If the number to the left is less than or equal to the number to the right, it returns true. If the number on the left is greater than the number on the right, it returns false. Like the equality operator, the less than or equal to operator converts data types.

**Examples**

4 <= 5

'7' <= 7

5 <= 5

3 <= 2

'8' <= 4

In order, these expressions would evaluate to true, true, true, false, and false.

Add the less than or equal to operator to the indicated lines so that the return statements make sense.

Hint-

* testLessOrEqual(0) should return the string Smaller Than or Equal to 12
* testLessOrEqual(11) should return the string Smaller Than or Equal to 12
* testLessOrEqual(12) should return the string Smaller Than or Equal to 12
* testLessOrEqual(23) should return the string Smaller Than or Equal to 24
* testLessOrEqual(24) should return the string Smaller Than or Equal to 24
* testLessOrEqual(25) should return the string More Than 24
* testLessOrEqual(55) should return the string More Than 24
* You should use the <= operator at least twice

**Code:**

**function testLessOrEqual(val) {**

**if (val <= 12) {**

**return "Smaller Than or Equal to 12";**

**}**

**if (val <= 24) {**

**return "Smaller Than or Equal to 24";**

**}**

**return "25 or More";**

**}**

**testLessOrEqual(10);**

Q7- **Comparisons with the Logical And Operator**

Sometimes you will need to test more than one thing at a time. The logical and operator (&&) returns true if and only if the operands to the left and right of it are true.

The same effect could be achieved by nesting an if statement inside another if:

if (num > 5) {

if (num < 10) {

return "Yes";

}

}

return "No";

will only return Yes if num is greater than 5 and less than 10. The same logic can be written as:

if (num > 5 && num < 10) {

return "Yes";

}

return "No";

Replace the two if statements with one statement, using the && operator, which will return the string Yes if val is less than or equal to 50 and greater than or equal to 25. Otherwise, will return the string No.

* You should use the && operator once
* You should only have one if statement
* testLogicalAnd(0) should return the string No
* testLogicalAnd(24) should return the string No
* testLogicalAnd(25) should return the string Yes
* testLogicalAnd(30) should return the string Yes
* testLogicalAnd(50) should return the string Yes
* testLogicalAnd(51) should return the string No
* testLogicalAnd(75) should return the string No
* testLogicalAnd(80) should return the string No

**Code:**

**function testLogicalAnd(val) {**

**if (val <= 50 && val >= 25) {**

**return 'Yes';**

**}**

**return "No";**

**}**

**testLogicalAnd(10);**

Q8-**Comparisons with the Logical Or Operator**

The logical or operator (||) returns true if either of the operands is true. Otherwise, it returns false.

The logical or operator is composed of two pipe symbols: (||). This can typically be found between your Backspace and Enter keys.

The pattern below should look familiar from prior waypoints:

if (num > 10) {

return "No";

}

if (num < 5) {

return "No";

}

return "Yes";

will return Yes only if num is between 5 and 10 (5 and 10 included). The same logic can be written as:

if (num > 10 || num < 5) { return "No";

}

return "Yes";

Combine the two if statements into one statement which returns the string Outside if val is not between 10 and 20, inclusive. Otherwise, return the string Inside.

Hint

* You should use the || operator once
* You should only have one if statement
* testLogicalOr(0) should return the string Outside
* testLogicalOr(9) should return the string Outside
* testLogicalOr(10) should return the string Inside
* testLogicalOr(15) should return the string Inside
* testLogicalOr(19) should return the string Inside
* testLogicalOr(20) should return the string Inside
* testLogicalOr(21) should return the string Outside
* testLogicalOr(25) should return the string Outside

**Code:**

**function testLogicalOr(val) {**

**if (val >10 || val < 20) {**

**return 'Inside';**

**}**

**return "Outside";**

**}**

**testLogicalOr(10);**

Q9-**Introducing Else Statements**

When a condition for an if statement is true, the block of code following it is executed. What about when that condition is false? Normally nothing would happen. With an else statement, an alternate block of code can be executed.

if (num > 10) {

return "Bigger than 10";

} else {

return "10 or Less";

}

Combine the if statements into a single if/else statement.

* You should only have one if statement in the editor
* You should use an else statement
* testElse(4) should return the string 5 or Smaller
* testElse(5) should return the string 5 or Smaller
* testElse(6) should return the string Bigger than 5
* testElse(10) should return the string Bigger than 5
* You should not change the code above or below the specified comments.

**Code:**

**function testElse(val) {**

**if (val > 5) {**

**return “Bigger than 5”;**

**}**

**else { return “smaller than 5”;**

**}**

**}**

**testElse(4)**

Q10- **Introducing Else If Statements**

If you have multiple conditions that need to be addressed, you can chain if statements together with else if statements.

if (num > 15) {

return "Bigger than 15";

} else if (num < 5) {

return "Smaller than 5";

} else {

return "Between 5 and 15";

}

Convert the logic to use else if statements.

Hint-

* You should have at least two else statements
* You should have at least two if statements
* You should have closing and opening curly braces for each if else code block.
* testElseIf(0) should return the string Smaller than 5
* testElseIf(5) should return the string Between 5 and 10
* testElseIf(7) should return the string Between 5 and 10
* testElseIf(10) should return the string Between 5 and 10
* testElseIf(12) should return the string Greater than 10

**Code:**

**function testElseif(val) {**

**if (val > 10) {**

**return “Greater than 10”;**

**} else if (val < 5) {**

**return “Smaller than 5”;**

**} else { return “Between 5 and 10”; }**

**}**

**testElseif(7);**

Q11-**Introducing Else If Statements**

If you have multiple conditions that need to be addressed, you can chain if statements together with else if statements.

if (num > 15) {

return "Bigger than 15";

} else if (num < 5) {

return "Smaller than 5";

} else {

return "Between 5 and 15";

}

Convert the logic to use else if statements.

Hint-

* You should have at least two else statements
* You should have at least two if statements
* You should have closing and opening curly braces for each if else code block.
* testElseIf(0) should return the string Smaller than 5
* testElseIf(5) should return the string Between 5 and 10
* testElseIf(7) should return the string Between 5 and 10
* testElseIf(10) should return the string Between 5 and 10
* testElseIf(12) should return the string Greater than 10

**Code:**

**function testElseif(val) {**

**if (val > 10) {**

**return “Greater than 10”;**

**} else if (val < 5) {**

**return “Smaller than 5”;**

**} else { return “Between 5 and 10”; }**

**}**

**testElseif(7);**

Q12- **Logical Order in If Else Statements**

Order is important in if, else if statements.

The function is executed from top to bottom so you will want to be careful of what statement comes first.

Take these two functions as an example.

Here's the first:

function foo(x) {

if (x < 1) {

return "Less than one";

} else if (x < 2) {

return "Less than two";

} else {

return "Greater than or equal to two";

}

}

And the second just switches the order of the statements:

function bar(x) {

if (x < 2) {

return "Less than two";

} else if (x < 1) {

return "Less than one";

} else {

return "Greater than or equal to two";

}

}

While these two functions look nearly identical if we pass a number to both we get different outputs.

foo(0)

bar(0)

foo(0) will return the string Less than one, and bar(0) will return the string Less than two.

Change the order of logic in the function so that it will return the correct statements in all cases.

Hint-

* orderMyLogic(4) should return the string Less than 5
* orderMyLogic(6) should return the string Less than 10
* orderMyLogic(11) should return the string Greater than or equal to 10

**Code:**

**function orderMyLogic(val) {**

**if (val < 5) {**

**return "Less than 5";**

**} else if (val < 10) {**

**return "Less than 10";**

**} else {**

**return "Greater than or equal to 10";**

**}**

**}**

**orderMyLogic(7);**

Q13-**Chaining If Else Statements**

if/else statements can be chained together for complex logic. Here is pseudocode of multiple chained if / else if statements:

if (condition1) {

statement1

} else if (condition2) {

statement2

} else if (condition3) {

statement3

. . .

} else {

statementN

}

Write chained if/else if statements to fulfill the following conditions:

num < 5 - return Tiny

num < 10 - return Small

num < 15 - return Medium

num < 20 - return Large

num >= 20 - return Huge

Hint-

* You should have at least four else statements
* You should have at least four if statements
* You should have at least one return statement
* testSize(0) should return the string Tiny
* testSize(4) should return the string Tiny
* testSize(5) should return the string Small
* testSize(8) should return the string Small
* testSize(10) should return the string Medium
* testSize(14) should return the string Medium
* testSize(15) should return the string Large
* testSize(17) should return the string Large
* testSize(20) should return the string Huge
* testSize(25) should return the string Huge

**Code;**

**function testSize(num) {**

**if (num < 5) return "Tiny";**

**else if (num < 10) return "Small";**

**else if (num < 15) return "Medium";**

**else if (num < 20) return "Large";**

**else return "Huge";**

**}**

**testSize(7);**

Q14- **Golf Code**

In the game of [golf](https://en.wikipedia.org/wiki/Golf), each hole has a par, meaning, the average number of strokes a golfer is expected to make in order to sink the ball in the hole to complete the play. Depending on how far above or below par your strokes are, there is a different nickname.

Your function will be passed par and strokes arguments. Return the correct string according to this table which lists the strokes in order of priority; top (highest) to bottom (lowest):

| **Strokes** | **Return** |
| --- | --- |
| 1 | "Hole-in-one!" |
| <= par - 2 | "Eagle" |
| par - 1 | "Birdie" |
| par | "Par" |
| par + 1 | "Bogey" |
| par + 2 | "Double Bogey" |
| >= par + 3 | "Go Home!" |

par and strokes will always be numeric and positive. We have added an array of all the names for your convenience.

Hint-

* golfScore(4, 1) should return the string Hole-in-one!
* golfScore(4, 2) should return the string Eagle
* golfScore(5, 2) should return the string Eagle
* golfScore(4, 3) should return the string Birdie
* golfScore(4, 4) should return the string Par
* golfScore(1, 1) should return the string Hole-in-one!
* golfScore(5, 5) should return the string Par
* golfScore(4, 5) should return the string Bogey
* golfScore(4, 6) should return the string Double Bogey
* golfScore(4, 7) should return the string Go Home!
* golfScore(5, 9) should return the string Go Home!

**Code:**

**function golfScore(par, strokes) {**

**if (strokes === 1) return "Hole-in-one!";**

**else if ((strokes - par) <= -2) return 'Eagle';**

**else if ((strokes - par) === -1) return 'Birdie';**

**else if (strokes === par) return 'Par';**

**else if ((strokes - par) === 1) return 'Bogey';**

**else if ((strokes - par) === 2) return 'Double Bogey';**

**else return 'Go Home!';**

**}**

**golfScore(5, 1);**

Q15-**Selecting from Many Options with Switch Statements**

If you have many options to choose from, use a switch statement. A switch statement tests a value and can have many case statements which define various possible values. Statements are executed from the first matched case value until a break is encountered.

Here is an example of a switch statement:

switch(lowercaseLetter) {

case "a":

console.log("A");

break;

case "b":

console.log("B");

break;

}

case values are tested with strict equality (===). The break tells JavaScript to stop executing statements. If the break is omitted, the next statement will be executed.

Write a switch statement which tests val and sets answer for the following conditions:

1 - alpha

2 - beta

3 - gamma

4 - delta

Hint-

* caseInSwitch(1) should have a value of the string alpha
* caseInSwitch(2) should have a value of the string beta
* caseInSwitch(3) should have a value of the string gamma
* caseInSwitch(4) should have a value of the string delta
* You should not use any if or else statements
* You should have at least 3 break statements

**Code:**

**function caseInSwitch(val) {**

**var answer = "";**

**switch (val) {**

**case 1:**

**answer = "alpha";**

**break;**

**case 2:**

**answer = "beta";**

**break;**

**case 3:**

**answer = "gamma";**

**break;**

**case 4:**

**answer = "delta";**

**break;**

**}**

**return answer;**

**}**

**caseInSwitch(1);**

Q16-**Adding a Default Option in Switch Statements**

In a switch statement you may not be able to specify all possible values as case statements. Instead, you can add the default statement which will be executed if no matching case statements are found. Think of it like the final else statement in an if/else chain.

A default statement should be the last case.

switch (num) {

case value1:

statement1;

break;

case value2:

statement2;

break;

...

default:

defaultStatement;

break;

}

Write a switch statement to set answer for the following conditions:

a - apple

b - bird

c - cat

default - stuff

Hint-

* switchOfStuff("a") should return the string apple
* switchOfStuff("b") should return the string bird
* switchOfStuff("c") should return the string cat
* switchOfStuff("d") should return the string stuff
* switchOfStuff(4) should return the string stuff
* You should not use any if or else statements
* You should use a default statement
* You should have at least 3 break statements

**Code:**

**function switchOfStuff(val) {**

**var answer = "";**

**switch (val) {**

**case "a":**

**answer = "apple";**

**break;**

**case "b":**

**answer = "bird";**

**break;**

**case "c":**

**answer = "cat";**

**break;**

**default:**

**answer = "stuff";**

**}**

**return answer;**

**}**

**switchOfStuff(1);**

Q17- **Multiple Identical Options in Switch Statements**

If the break statement is omitted from a switch statement's case, the following case statement(s) are executed until a break is encountered. If you have multiple inputs with the same output, you can represent them in a switch statement like this:

var result = "";

switch(val) {

case 1:

case 2:

case 3:

result = "1, 2, or 3";

break;

case 4:

result = "4 alone";

}

Cases for 1, 2, and 3 will all produce the same result.

Write a switch statement to set answer for the following ranges:

1-3 - Low

4-6 - Mid

7-9 - High

**Note:** You will need to have a case statement for each number in the range.

Hint-

* sequentialSizes(1) should return the string Low
* sequentialSizes(2) should return the string Low
* sequentialSizes(3) should return the string Low
* sequentialSizes(4) should return the string Mid
* sequentialSizes(5) should return the string Mid
* sequentialSizes(6) should return the string Mid
* sequentialSizes(7) should return the string High
* sequentialSizes(8) should return the string High
* sequentialSizes(9) should return the string High
* You should not use any if or else statements
* You should have nine case statements

**Code:**

**function sequentialSizes(val) {**

**var answer = "";**

**switch (val) {**

**case 1:**

**case 2:**

**case 3:**

**answer = "Low";**

**break;**

**case 4:**

**case 5:**

**case 6:**

**answer = "Mid";**

**break;**

**case 7:**

**case 8:**

**case 9:**

**answer = "High";**

**break;**

**}**

**return answer;**

**}**

**sequentialSizes(1);**

Q18- **Replacing If Else Chains with Switch**

If you have many options to choose from, a switch statement can be easier to write than many chained if/else if statements. The following:

if (val === 1) {

answer = "a";

} else if (val === 2) {

answer = "b";

} else {

answer = "c";

}

can be replaced with:

switch(val) {

case 1:

answer = "a";

break;

case 2:

answer = "b";

break;

default:

answer = "c";

}

Change the chained if/else if statements into a switch statement.

* You should not use any else statements anywhere in the editor
* You should not use any if statements anywhere in the editor
* You should have at least four break statements
* chainToSwitch("bob") should be the string Marley
* chainToSwitch(42) should be the string The Answer
* chainToSwitch(1) should be the string There is no #1
* chainToSwitch(99) should be the string Missed me by this much!
* chainToSwitch(7) should be the string Ate Nine
* chainToSwitch("John") should be "" (empty string)
* chainToSwitch(156) should be "" (empty string)

**Code:**

**function chainToSwitch(val) {**

**var answer = “”;**

**switch (val) {**

**case "bob":**

**answer="Marley";**

**break;**

**case '42':**

**answer="The Answer";**

**break;**

**case '1':**

**answer="There is no #1";**

**break;**

**case '99':**

**answer="Missed me by this much!";**

**break;**

**case '7':**

**answer="Ate Nine";**

**break;**

**default:**

**answer="default";**

**break;**

**}**

**return answer;**

**}**

**chainToSwitch(100);**

Q19- **Returning Boolean Values from Functions**

You may recall from [Comparison with the Equality Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-equality-operator) that all comparison operators return a boolean true or false value.

Sometimes people use an if/else statement to do a comparison, like this:

function isEqual(a,b) {

if (a === b) {

return true;

} else {

return false;

}

}

But there's a better way to do this. Since === returns true or false, we can return the result of the comparison:

function isEqual(a,b) {

return a === b;

}

Fix the function isLess to remove the if/else statements.

Hint-

* isLess(10,15) should return true
* isLess(15,10) should return false
* You should not use any if or else statements

**Code:**

**function isLess(a, b) {**

**return a < b;**

**}**

**isLess(10, 15);**

Q20-**Return Early Pattern for Functions**

When a return statement is reached, the execution of the current function stops and control returns to the calling location.

**Example**

function myFun() {

console.log("Hello");

return "World";

console.log("byebye")

}

myFun();

The above will display the string Hello in the console, and return the string World. The string byebye will never display in the console, because the function exits at the return statement.

Modify the function abTest so that if a or b are less than 0 the function will immediately exit with a value of undefined.

**Hint**

Remember that [undefined is a keyword](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/understanding-uninitialized-variables), not a string.

Hint-

* abTest(2,2) should return a number
* abTest(2,2) should return 8
* abTest(-2,2) should return undefined
* abTest(2,-2) should return undefined
* abTest(2,8) should return 18
* abTest(3,3) should return 12
* abTest(0,0) should return 0

**Code:**

**function abTest(a, b) {**

**if (a < 0 || b < 0) {**

**return undefined;**

**}**

**return Math.round(Math.pow(Math.sqrt(a) + Math.sqrt(b), 2));**

**}**

**abTest(2, 2);**