OPERATORS:

Task 1: Using the logical && operator

You are coding an RPG game, where each character has certain skill levels based on the value saved in their score.

1. Create a variable named `score` and set it to `8`
2. Use `console.log()` that includes the string `"Mid-level skills:"` and compares the `score` variable to above `0` and below `10` using the `&&` operator

The expected output in the console should be: `"Mid-level skills: true."`

Task 2: Using the logical || operator

Imagine you are coding a video game. Currently, you're about to code some snippets related to the game over condition.

You need to create a new variable named "timeRemaining" and set it to `0`. You also need to code a new variable named "energy" and set it to `10`.

Next, you should write a piece of code that could be used to determine if the game is over, based on whether either the value of the `timeRemaining` variable is `0` or the value of the `energy` variable is `0`.

Complete the task using the following steps:

1. Declare the variable `timeRemaining`, and assign the value of `0` to it.
2. Declare the variable `energy`, and assign the value of `10` to it.
3. Console log the following parameters: `"Game over: "`, and `timeRemaining == 0 || energy == 0`

Note that the expected output in the console should be: `"Game over: true."`

Task 3: Using the modulus operator, `%`, to test if a given number is odd

You need to code a small program that takes a number and determines if it's an even number (like 2, 4, 6, 8, or 10).

To achieve this task, you need to declare six variables, as follows:

1. The first variable, named `num1,` should be assigned a number value of `2`.
2. The second variable, named `num2`, should be assigned a number value of `5`.
3. The third variable, named `test1`, should be assigned the calculation of `num1 % 2`.
   Note: executing this code will return a number.

4. The fourth variable, named `test2`, should be assigned the calculation of `num2 % 2`. **Note**: executing this code will also return a number.
5. The fifth variable, named `result1`, should be assigned the result of comparing if the number stored in the `test1` variable is not equal to `0`, in other words, this: `test1 == 0`.
6. The sixth variable, named `result2`, should be assigned the result of comparing if the number stored in the `test2` variable is not equal to `0`, in other words, `test2 == 0`.

Run console.log two times after you've set the variables:

1. The first console log should have the following code between parentheses: `"Is", num1, "an even number?", result1`
2. The second console log should have the following code between parentheses: `"Is", num2, "an even number?", result2`

Note: The output to the console should be as follows:

```
Is 2 an even number? true

Is 5 an even number? false
```

Task 4: Add numbers using the + operator

Console log the result of adding two numbers, `5` and `10`, using the `+` operator.

Note: This task should be completed on a single line of code. The output in the console should be `15`.

Task 5: Concatenate numbers and strings using the + operator

Code three variables:

1. The first variable should be a string with the following value: `"Now in "`. Name the variable `now`.
2. The second variable should be a number with the value: `3`. Name the variable `three`.
3. The third variable should be a string with the following value: `"D!"`. Name the variable `d`.
4. Console log the following code: `now + three + d`.

Note: The expected output should be: `"Now in 3D!"`.

Task 6: Use the `+=` operator to accumulate values in a variable

Code a new variable and name it `counter`, assigning it a value of `0`.

On the next line, use the `+=` operator to increase the value of the counter by `5`.

On the next line, use the `+=` operator to increase the value of the counter by `3`.

On the fourth line, console log the value of the `counter` variable.

Note: The output value should be `8`.

CONDITIONALS:

Introduction

In this exercise, you will practice working with `if else` statements. By the end of this exercise, you will be able to write an "`if else`" statement that determines your source of income based on your age. You will also be able to write a `switch` statement that determines your evening routine based on the day of the week.

Complete the following steps to create: Are You Old Enough?

1. Declare a variable age using the `const` keyword and set it to the number **10**.
2. Add an `if` statement that checks if the value of the `age` variable is greater than or equal to the number **65**. Inside the `if` block, console.log the sentence: "You get your income from your pension".
3. Add an "`else if`", where you'll check if the value of the age is less than **65** and greater than or equal to **18**. Inside this "`else if`" block, type "console.log" and then "Each month you get a salary".
4. Add another "`else if`", and this time check if the value of the age is under **18**. Inside the "`else if`" block, "type console.log" and then "You get an allowance".
5. Add an "`else`" statement to capture any other value. Inside the block, type "console.log" and then "The value of the age variable is not numerical".

Try adjusting the age and executing the program to see how it will affect the output.

Code the *days of the week* program as a switch statement

1. On the next line, define a new variable, name it `day`, and set its value to `"Sunday"`.
2. Start coding a `switch` statement, passing the `day` variable as the expression to evaluate.
3. Inside the `switch`, add cases for every day of the week, starting with 'Monday', and ending with 'Sunday'. Make sure to use string values for days. Inside each case, for now, just add a console.log('Do something'), and add a break; on the line below.
4. At the very bottom of the `switch` statement, add the default case and add a console.log('There is no such day').
5. Finally, update the console.log calls for each case, based on whatever activity you have on each of the days.

Tips

- If you need to make sure that multiple conditions are true in an if statement, you can do so using the `&&` operator
- In JavaScript, the correct syntax of the "greater than or equal to" operator is: `>=`.
- Don't forget to add a break at the very end of each case in a switch statement.

Exercise 2. Use the completed code from the previous task, but convert the conditionals to a switch statement.

When you code the solution, the output in the console should remain exactly the same as in the previous question.

Note: You'll need three separate cases for the three medals, and a default case for all other values of the `i` variable.

https://www.javascripttutorial.net/javascript-comparison-operators/

Practicing with functions

Your task in this exercise is to code a function that will be able to take a word and locate the position of a chosen letter in that given word.

Task 1:

Write a function named "letterFinder" that accepts two parameters: *word* and *match*.

Task 2:

Code a "for" loop inside the function's body. The for loop's counter should start at zero, increment by 1 on each iteration, and exit when the counter variable's value is equal to the length of the *word* parameter.

Task 3:

Add an if statement inside the for loop whose condition works as follows:

1. **Access each of the letters inside the passed in *word* using the counter variable, with *word[i]*.**
2. **Check if the current *word[i]* is equal to the value of *match*.**

Task 4:

console.log the following inside the body of the if statement: *console.log('Found the', match, 'at', i).*

Task 5:

Write the else condition. Here you'll just console log the following: *console.log('---No match found at', i).*

Task 6:

Call the *letterFinder* and pass it as its first argument the string *"test,"* and as its second argument, the string *"t."*

Your output should be the following:

Found the t at 0

---No match found at 1

---No match found at 2

Found the t at 3

JavaScript improvements brief history

In this reading, you will learn about the history of JavaScript and the importance of ECMA (European Computer Manufacturers Association) and ECMAScript.

JavaScript is a programming language that had humble beginnings.

It was built in only 10 days in 1995 by a single person, Brendan Eich, who was tasked with building a simple scripting language to be used in version 2 of the Netscape browser. It was initially called LiveScript, but since the Java language was so popular at the time, the name was changed to JavaScript - although Java and JavaScript are in no way related.

For the first few years, after it was built, JavaScript was a simple scripting language to add mouseover effects and other interactivity. Those effects were being added to webpages using the `<script>` HTML element.

Inside each of the script elements, there could be some JavaScript code. Due to the rule that HTML, CSS, and JavaScript must be backward compatible, even the most advanced code written in JavaScript today ends up being written between those script tags.

Over the years, JavaScript grew ever more powerful, and in recent times, it's continually touted as among the top three commonly used languages.

In 1996 Netscape made a deal with the organization known as ECMA (European Computer Manufacturers Association) to draft the specification of the JavaScript language, and in 1997 the first edition of the ECMAScript specification was published.

ECMA publishes this specification as the ECMA-262 standard.

You can think of a standard as an agreed-upon way of how things should work. Thus, ECMA-262 is a standard that specifies how the JavaScript language should work.

There have been 12 ECMA-262 updates - the first one was in 1997.

JavaScript as a language is not a completely separate, stand-alone entity. It only exists as an implementation. This implementation is known as a JavaScript engine.

Traditionally, the only environment in which it was possible to run a JavaScript engine, was the browser. More specifically, a JavaScript engine was just another building block of the browser. It was there to help a browser accomplish its users' goal of utilizing the internet for work, research, and play.

So, when developers write JavaScript code, they are using it to interact with a JavaScript engine. Put differently, developers write JavaScript code so that they can "talk to" a JavaScript engine.

Additionally, the JavaScript engine itself comes with different ways to interact with various other parts of the browser. These are known as Browser APIs.

Thus, the code that you write in the JavaScript programming language allows you to: 1. Interact with the JavaScript engine inside of the browser 2. Interact with other browser functionality that exists outside of the JavaScript engine, but is still inside the browser.

Although traditionally it was possible to interact with the JavaScript engine only inside of the browser, this all changed in 2009, when Node.js was built by Ryan Dahl.

He came up with a way to use a JavaScript engine as a stand-alone entity. Suddenly, it was possible to use JavaScript outside of the browser, as a separate program on the command line, or as a server-side environment.

Today, JavaScript is ubiquitous and is running in browsers, on servers, actually, on any device that can run a JavaScript engine.

https://insights.stackoverflow.com/survey/2021#most-popular-technologies-op-sys-prof

The above shows ratings of programming language

https://developer.mozilla.org/en-US/docs/Glossary/Primitive

Further reading on javascript primitive values