TASK

Logical Programming — Operations and Switch Statements

Visit our website

Introduction

WELCOME TO THE OPERATORS AND SWITCH STATEMENTS TASK!

In a programming language, an operator is a symbol that tells the compiler or interpreter to perform specific operations (whether it be mathematical, relational or logical) and produce a final result. This task will introduce you to the different types of operators and show you how to use them. You will also be introduced to a more succinct version of if-statements: the switch statement!

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

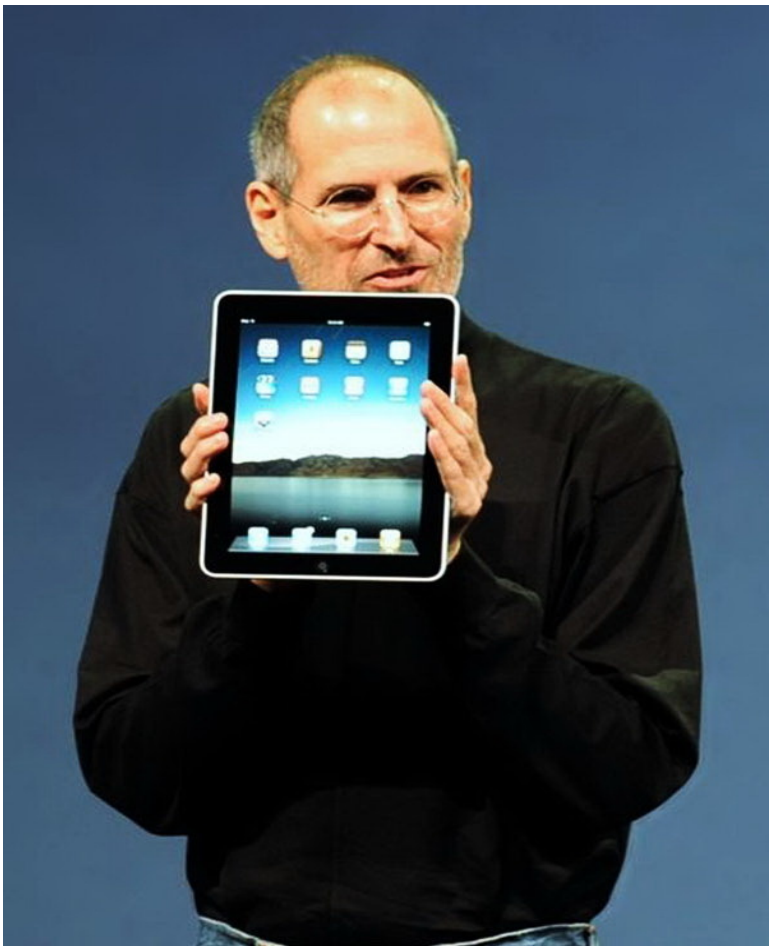
Our expert code reviewers are happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

A note from the

HyperionDev Team

The iPad combines many of the popular capabilities of the iPhone, such as built-in high-definition camera, access to the iTunes Store, and audio-video capabilities, but with a nine-inch screen and without the phone.

Apps, games, and accessories helped spur the popularity of the iPad and led to its adoption in thousands of different applications from movie making, creating art, making music, inventory control and point-of-sale systems, to name but a few.



WHAT ARE OPERATORS?

Operators are symbols that tell the computer which mathematical calculations to perform or which comparisons to make.

COMPARISON OPERATORS

Here is a quick reminder of our comparison operators:

The four basic comparative operators are:

- greater than >
- less than <
- equal to ==
- equal to in value and type ===
- not !

We can combine the greater than, less than and not operator with the equals operator and form three new operations.

- greater than or equal to >=
- less than or equal to <=
- not equal to !=

Comparing strings:

```
let myName = "Tom";
if (myName == "Tom") {
    console.log("I was looking for you");
}
```

Comparing numbers:

```
let num1 = 10;
let num2 = 20;

if (num1 >= num2) {       // The symbol for 'greater than or equal to' is >=
    console.log("It's not possible that 10 is bigger than or equal to 20.");
}
else if (num1 <= num2) {     //The symbol for 'less than or equal to' is <=
    print("10 is less than or equal to 20.");
}
else if (num1 != num2) {     // The symbol for 'not equal to' is !=
    console.log("This is also true since 10 isn't equal to 20, but the
else-if statement before comes first and is true so JavaScript will execute
that!");
```

```
}
else if (num1 == num2) {          // The symbol for 'equal to' is ==
    console.log("Will never execute this print statement...");
}
else if (num1 === num2) { // The symbol for 'equal to in value and type' is
===
    console.log("Will never execute this print statement either...");
}
```

The program will check the first part of the if statement (is **num1** bigger than or equal to **num2**?).

If it is not, then it goes into the first else-if statement and checks if **num1** is less than or equal to **num2**. If it is not then it goes into the next else-if statement, etc.

In the example above, the values of the variables (**num1** and **num2**) are hard-coded into the program in order to create a simple example for teaching purposes. Because of this the code will always evaluate, the same way, producing the same result. In reality, a programmer is more likely to be working with variables with values that are the result of user input or calculations that involve other variables. In such scenarios, the practical value of knowing how to create a series of conditional statements to evaluate the values of variables should be much clearer.

LOGICAL OPERATORS

Sometimes you may need to use logical operators: AND (**&&**), OR (**||**) or NOT (**!**). The AND and OR operators can be used to combine boolean expressions. The resulting expression will still evaluate to either true or false. For example, consider the code below:

```
let num = 12;
if (num >= 10 && num <= 15){
    console.log(num + " is a value between 10 and 15");
}
```

The boolean expression (**num >= 10 && num <= 15**) evaluates to true in the example above because both the expression **num >= 10** AND the expression **num <= 15** are true. This is an example of a conjunction operation where both conditions need to be true for the whole statement to be true. The symbol for the AND operator is **&&.**

To illustrate how the OR operator is used, consider a real-life example: you could buy a very nice car if you have enough money OR if someone gives you the money as a gift OR if you can get a loan.

```
let lotsOfMoney = false;
let receivedGift = false;
let loanApproved = true;

if (lotsOfMoney || receivedGift || loanApproved){
    console.log("Can purchase a car");
} else {
       console.log("Sorry! Can't afford a car");
}
```

This is a disjunction operation where at least one of the conditions needs to be true for the whole statement to be true. The boolean expression (`lotsOfMoney || receivedGift || loanApproved`) is true because at least one of the expressions that make up that compound expression is true. The `||` symbols are used for the OR operation.

Try this:

- Open the JavaScript Console.
- Copy and paste the code above into the console.
- Execute the code and take note of the output. Make sure you understand why your code produced the output it did.

THE SWITCH CONDITIONAL

Now that you are comfortable using conditional statements, we can simplify it into a switch statement. This structure allows multiple conditions to be evaluated with a much more efficient and structured layout than many ***else if*** statements. The syntax of the **switch** statement is given below:

```
switch(expression)
{
      case 1:
            Do something;
            break;
      case 2:
            Do something else;
            break;
      default:
            Else do this;
            break;
```

```
}
```

In the above example, the **break** statement is used to terminate the processing of each case inside the **switch** statement. It transfers the flow of control of the program to the next piece of code, which in the example above is the next case, until in the final case the **break** statement ends the execution of the **switch** statement. You can read more about break statements in JavaScript **here**.

Now, consider an example of the **switch** statement in action below. The statement in the example uses the *getDay* function to get the current day of the week. If `Date().getDay() == 0`, then the variable 'day' is assigned the value "Sunday", else if `Date().getDay() == 1`, then the variable 'day' is assigned the value "Monday" etc.

```
let day = new Date().getDay();
/*"new Date().getDay()" is built-in JavaScript code. To see more about how this works, visit this site:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getDay */
console.log("The value of day is " + day);
switch (day) {
    case 0:
       day = "Sunday";
        break;
    case 1:
       day = "Monday";
        break;
    case 2:
       day = "Tuesday";
        break;
    case 3:
       day = "Wednesday";
        break;
    case 4:
       day = "Thursday";
        break;
    case 5:
       day = "Friday";
        break;
    case 6:
       day = "Saturday";
}
console.log("Today is " + day);
```

Instructions

Open **example.js** in Visual Studio Code and read through the comments before attempting these tasks.

Getting to grips with JavaScript takes practice. You will make mistakes in this task. This is completely to be expected as you learn the keywords and syntax rules of this programming language. It is vital that you learn to debug your code. To help with this remember that you can:

- Use either the JavaScript console or Visual Studio Code (or another editor of your choice) to execute and debug JavaScript in the next few tasks.
- Remember that if you really get stuck, you can contact an expert code reviewer for help.

Compulsory Task 1

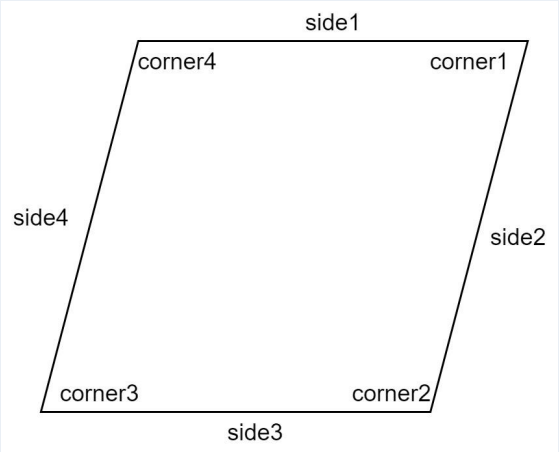
Follow these steps:

- *Note: For this task, you will need to create an HTML file to get input from a user. If you need a refresher on how to do this, go back to the **example.js** and **index.html** files in your Task 2 folder for a refresher.*
- Create a new JavaScript file in this folder called **seven11.js**.
- Create a program to determine if a number input by the user is divisible by 7 and 11, either 7 or 11, or neither.
- Output the answer, depending on the result, with one of the following statements:
    - `“[input] is divisible by both 7 and 11.”`
    - `“[input] is divisible by 11.”`
    - `“[input] is divisible by 7.”`
    - `“[input] is divisible by neither 7 nor 11.”`

Compulsory Task 2

Follow these steps:

- Create a new JavaScript file in this folder called **quadrilaterals.js**.
- There are multiple quadrilaterals, all with their own properties. Create a program that receives the length of each side and the angles of each corner (these can be hard-coded) and determines whether the shape is a square, a rectangle, a rhombus or a parallelogram.
- The properties of each are as follows:
    - **Square:** All sides equal, all corners 90 degrees.
    - **Rectangle:** Opposite sides equal, all corners 90 degrees.
    - **Rhombus:** All sides equal, opposite angles equal. Two opposite corners are less than 90 degrees, the other two corners are more than 90 degrees.
    - **Parallelogram:** Opposite sides equal, opposite angles equal. Two opposite corners are less than 90 degrees, the other two corners are more than 90 degrees.
- When labelling the variables for your program, use this diagram below as a guide:



- For more information on quadrilaterals, you can have a look **here**.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.