TASK

Beginner Control Structures — if, else, and else-if Statements

Visit our website

Introduction

WELCOME TO THE BEGINNER CONTROL STRUCTURES — IF, ELSE AND ELSE-IF STATEMENTS TASK!

In this task, you will learn about a program's flow control. A control structure is a block of code that analyses variables and chooses a direction in which to go based on given parameters. In essence, it is a decision-making process in computing that determines how a computer responds to certain conditions.

A note from the

HyperionDev Team

Let's consider how we make decisions in real life. When you are faced with a problem, you have to see what the problem entails. Once you figure out the crux of the problem, you then follow through with a way to solve this problem. Teaching a computer how to solve problems works in a similar way. We tell the computer to look out for a certain problem and tell it how to solve the problem when faced with it.

What is Computational Thinking?
Computational Thinking is the process of solving a problem using a certain manner of thinking. Computational Thinking is used in the development of computer applications, but it can also be used to solve problems across a wide variety of categories, including math, science, geography and art. After learning about Computational Thinking, you will be able to apply this to work as well as real-life problems.

Simple Daily Examples of Computational Thinking

- Looking up a name in your contact list
- Cooking a gourmet meal

IF STATEMENTS

We are now going to learn about a vital concept when it comes to programming. We will be teaching the computer how to make decisions for itself using an *if statement*. As the name suggests, it is essentially a question, a way of comparing two or more variables or scenarios and performing a specified action based on the outcome of that comparison.

If statements contain a condition. The condition is the part of the *if statement* in brackets in the example below. Conditions are statements that can only evaluate to true or false. If the condition is true, then the indented statements are executed. If the condition is false, then the indented statements are skipped. As such, *if* statements, and the *else* and *else-if* constructs you are going to learn about soon, are all what we call "conditional statements".

In JavaScript, *if statements* have the following general syntax:

```
if (condition) {
        indented statements;
}
```

Here's an example of a JavaScript *if statement*:

```
let num = 10;

if (num < 12) {
console.log("the variable num is lower than 12");
}
```

This *if statement* checks if the variable number is less than 12. If it is, then it will output the sentence letting us know. If *num* were greater than 12, then it would not output that sentence.

Notice the following important syntax rules for an *if statement*:

- In JavaScript, the opening curly bracket is followed by code that can only run if the statement's condition is true.
- The closing curly bracket shows the end of the statement that will execute if the statement's condition is true.

COMPARISON OPERATORS

You may have also noticed the less than (<) symbol above. As a programmer, it's important not to forget the basic logical commands. We use comparison operators

to compare values or variables in programming. These operators work well with *if statements* and *loops* to control what goes on in our programs.

Operator	Description	Example
>	greater than	Condition: 12 > 1 Result: True
<	less than	Condition: 12 < 1 Result: False
>=	greater than or equal to	Condition: 12 >= 1 Result: True
<=	less than or equal to	Condition: 12 <= 12 Result: True
==	equals	Condition: 12 == 1 Result: False
===	Equal value and equal type	Condition: 12 === “twelve” Result: False
!=	does not equal	Condition: 12 != 1 Result: True

Take note that the symbol we use for equal to is ‘==’, and not ‘=’. This is because ‘==’ literally means ‘equals’ (e.g. *i == 4* means *i* is equal to 4). We use ‘==’ in conditional statements to check if the values are the same. Similarly, we use ‘===’ in conditional statements to check if the value *and* type are the same (e.g. 5 === 5 would return true, but 5 === “5” would return false).

On the other hand, ‘=’ is used to assign a value to a variable (e.g. *i = “blue”* means I assign the value of *“blue”* to *i*). This is a subtle but important difference. You’re welcome to check back in the previous tasks if you feel like you need a refresher on variables at this point.

A note from our coding mentor

Seraaj

Sorry to interrupt but I found this fantastic blog written by MARC CHERNOFF that shows 10 if Statements in ”Real Life” that are worth learning. It just shows how important making

decisions is to our lives. It also shows you that if you don't follow the desired path based on your decision, the outcome could be detrimental to you.

1. *If you don't understand the product or service, don't buy it until you do.*
2. *If you do not take ownership of your actions, your actions will eventually own you.*
3. *If you are not saving at least 10% of your salary, you are not saving enough.*
4. *If you talk too much, people will stop listening. If you don't talk enough, people will never hear your point of view.*
5. *If you are lazy, you will fail. Laziness will always overshadow your true potential.*
6. *If you hate your job, you also hate half of the time you spend on this planet.*
7. *If you are not investing (120 minus your age) percent of your savings in the stock market, you are giving up thousands of dollars over the course of your lifetime.*
8. *If you don't finish what you start, your success rate will always be zero.*
9. *If you don't consume enough liquids, you will never be healthy.*
10. *If your monthly debt payments exceed 40% of your total income, you will go broke if you don't fix your spending habits promptly.*

ELSE STATEMENTS

If statements are one of the most important concepts in programming, but on their own, they are a bit limited. The *else statement* represents an alternative path for the flow of logic if the condition of the if statement turns out to be *false*.

Imagine if you were hungry and you sent your friend to the shop to buy chocolate. When they get to the shop, they find no chocolates and just leave because you didn't tell them any alternatives. They would have to keep coming back for instructions every time they didn't find what you wanted unless you provided them with an alternative. Similarly, instead of us having many 'if' statements to test each scenario, we can add an 'else' statement to give us a single alternative.

In JavaScript, the general if-else syntax is:

```
if (condition) {
    indented statements;
}
else {
    indented statements;
}
```

If the condition turns out to be *false*, the statements in the indented block following the *if* statement are skipped and the statements in the indented block following the *else* statement are executed.

Take a look at the following example:

```
let num = 10;
if (num < 12) {
    console.log("the variable num is lower than 12");
}
else {
    console.log("the variable num is greater than or equal to 12");
}
```

Now instead of nothing happening if the condition of the if statement is *False* (*num* ends up being greater than or equal to 12), the else statement will be executed.

Another example of using an else statement with an if statement can be found below. The value that the variable *hour* holds determines which string is assigned to *greeting*.

```
if (hour < 18) {
     greeting = "Good morning";
}
else {
     greeting = "Good evening";
}
```

We are faced with decisions like this on a daily basis. For instance, if it is cold outside you would likely wear a jacket. However, if it is not cold you might not find a jacket necessary. This is a type of branching. If one condition is true, do one thing and if the condition is false, do something else. This type of branching decision making can be implemented in JavaScript using 'if-else' statements.

ELSE IF STATEMENTS

The last piece of the puzzle when it comes to if statements is called an *else-if statement*. With this statement, we can add more conditions to the if statement, making it possible to test multiple parameters in the same statement.

Unlike the *else statement*, you can have multiple *else-if* statements in an if/else-if/else statement. If the condition of the if statement is *false*, the condition of

the next else-if statement is checked. If the first else-if statement condition is also *false*, the condition of the next else-if statement is checked, etc. If all the else-if conditions are *false*, the *else statement* and its indented block of statements are executed.

In JavaScript, *if/else-if/else statements* have the following syntax:

```
if (condition1) {
      indented statements;
}
else if (condition2) {
      indented statements;
}
else if (condition3) {
      indented statements;
}
else if (condition4) {
      indented statements;
}
else {
      indented statements;
}
```

Look at the following example:

```
let num = 10;

if (num >12) {
    console.log("the variable num is greater than 12");
}
else if (num > 10) {
    console.log("the variable num is greater than 10");
}
else if (num < 10) {
    console.log("the variable num is less than 10");
}
else {
    console.log("the variable num is 10");
}
```

Remember that you can combine if, else and else-if into one big statement.

Some important points to note on the syntax of if/else-if/else statements:

- Make sure that the if/else-if/else statements end with an open curly bracket and the code in each block ends with a closing curly bracket.
- Ensure that your indentation is correct (i.e. statements that are part of a certain control structure's 'code block' need the same indentation).
- To have an *else if* you must have an *if* above it.
- To have an *else* you must have an *if* or *else if* above it.
- You can't have an *else* without an *if* — think about it!
- You can have many *else if* statements under an *if*, but only one *else* right at the bottom. It's like the fail-safe statement that executes if the other *if/else if* statements fail!

NESTED IF STATEMENTS

We can also nest an if statement inside another if statement.

```
if (condition1){
    indented statements;

    //Nested if statement
    if (condition2) {
        indented statements;
    }
}
else{
    indented statements;
}
```

Look at the following example:

```
if (num>10){
    // Nested if statement
    if (num % 2 == 0) {
        console.log("num is larger than 10 and an even number.");
    }
    else{
        console.log("num is larger than 10 and an odd number.");
    }
}
else
```

```
{
    console.log("num is smaller or equal to 10.");
}
```

A note from our coding mentor
Ridhaa

Have you heard about Margaret Hamilton? This is the woman and engineer who took us to the moon. She wrote the code for Apollo 11's onboard flight software and, as a result of her work, she received NASA's Exceptional Space Act Award. If that's not enough, she is also credited with coining the term "software engineering". In the picture below, you will see a young Margaret standing next to the actual code she wrote to take humanity to the moon.

Margaret Hamilton

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 to be fully 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 1 folder.*
- Create a JavaScript file called **characters.js** in this folder.
- Ask the user to input an uppercase letter, a lowercase letter or a number.
- If the character is an uppercase letter, output their character and "is an uppercase letter." E.g. **`"R is an uppercase letter."`**
- If the character is a lowercase letter, output their character and "is a lowercase letter." E.g. **`"g is a lowercase letter."`**
- If the character is a number, output their character and "is a number." E.g. **`"6 is a number."`**
- If the character is none of these, output their character and "is not a letter or number". E.g. **`"? is not a letter or number."`**
- *Hint 1 : you may want to research the functions **`toUpperCase()`**, **`toLowerCase()`** and **`Number.isInteger()`***
- *Hint 2 : A Special character to upper case is equal a special character to lower case.*

Compulsory Task 2

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 1 folder for a refresher.*
- Create a JavaScript file called **waterTariffs.js** in this folder.
- These are the level 3 water tariffs for the city of Cape Town taken from **here**:

Water Steps (1kl = 1000 litres)	**Level 3 (2018/19)** Until 30/06/2019 **Rands (incl VAT)**
Step 1 (0 ≤ 6kl)	R15.73 (free for indigent households)
Step 2 (>6 ≤ 10.5kl)	R22.38 (free for indigent households)
Step 3 (>10.5 ≤ 35kl)	R31.77
Step 4 (>35kl)	R69.76

- The table above states that the first 6 000 litres will cost R15.73 per kilolitre. Next, water consumption above 6 000 litres but less than or equal to 10 500 litres will be charged at R22.38 per kilolitre. Therefore, a household that has used 8000 litres will pay R139.14 (15.73 x 6 + 22.38 x 2). The table carries on in this manner.
- Create a calculator to determine your water bill.
- The calculator should ask the user to input the number of litres of water they have used, and it should output the total amount in Rands (R) that they need to pay.

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.