Exercise: Working with Conditionals

Conditionals

Comparison Operators

As seen in earlier examples, comparison operators are used to compare two variables. The result of this comparison is either a `true` or `false`. Here's a list of available comparison operators:

Operator	Description	Example	Result
==	equal to	"1 == 1	true
===	equal value and equal type	“1” === 1	false
!=	not equal	1 != 2	true
!==	not equal value or not equal type	1 != 1	false
>	greater than	1 > 2	false
<	less than	1 < 2	true
>=	greater than or equal to	1 >= 1	true
<=	less than or equal to	2 <= 1	false

Conditionals

We use conditional statements to build logic into our code. The `if` statement executes a block of code if the condition is `truthy`, which means it evaluates true.

If the condition is `falsy`, it will run another block of code.

The result of a conditional comparison is always one of two values: `true` or `false`. This data type is called a `Boolean`.

Review the example below that compares score results:

```
function checkTestScore(score){
    let result;
    //ask if score is greater than 6
    if(score > 6){
        result = 'Advanced'
    }else{
        result = 'Moderate'
    }
    return result;
}
console.log(checkTestScore(8)) //expected 'Advanced'
console.log(checkTestScore(4)) // expected 'Moderate'
```

As you can see, the conditional statement checks if a score is great than 6. If the score is greater than 6, the result is Advanced. If the score is less than 6, the result is Moderate.

Multiple Conditions

Multiple `if-else` statements can be nested to create an `else if` clause. This can be helpful when you have multiple conditions to check before executing certain code.

```
var money = 10;
var Hungry = true;

function getFood(money) {
  if(money > 10) {
        if(Hungry === true)
            console.log("You can go out to eat!");
        else
            console.log("You can save money!");
  else {
    console.log("You're eating ramen again...");
  }
}
```

In the example above there are multiple conditions being considered. We're checking that `money` is greater than 10. If this is the case, then we're further checking that `hungry` is true. This is an example of nested `if-else` statements. The check for "is Hungry = true" will only happen is "money is greater than 10".

A note on semicolon

In JavaScript, you sometimes need to add a semicolon at the end of a code statement. The semicolon is only required when you have two or more statements on the same line. For example:

```
var i = 0; i++    // <-- semicolon required for first statement, but not the second one since this one
```

The semicolon in JavaScript is used to separate statements, but it can be omitted if the statement is followed by a line break (or there's only one statement in a {block}). For example, the following code is correct:

```
var i = 0
var j = 1
```

Task Instructions

Your task in this activity is to create a function that checks if a person is old enough to vote by checking their age. This function is called `isOldEnoughToVote(age)` and has the following specifications:

It takes an argument called `age` representing the age of the person. It checks if the age is greater than or equal to 18. If returns `true` or `false` based on that comparison.

Task

- Given a `functionisOldEnoughToVote`() with an age argument, return a string that checks if the age passed in is old enough to vote.