Javascript Conditional Statements and Logical Operators

Background Story

Odessa, an IT student in the Philippines is also a student council member 🏛. One of her responsibilities is to manage the student council budget. As the year progresses, she begins to notice discrepancies in the entries, especially after events 🎉. There are entries that are either too high or too low to be standard expenditures.

To simplify her work 🤷 and make everything more efficient, Odessa decides to write a script to filter these entries. Her plan is to write code that checks each entry and determines if it's above or below the standard limit. This is her first encounter with conditional statements, and she's excited to learn how to use them to make smart decisions in her code.

With your help, she will learn how to write 'if' and 'else' statements, how to use comparison operators (>, <, >=, <=, ==, !=), and logical operators (&&, ||, !) in JavaScript 💻.

Theory & Lecture Content

JavaScript if...else statement: Control structures are used to control the flow of execution in a program. The *if* statement is the simplest form of control structure. It evaluates whether a statement is true or false, and then runs a set of statements based on the result. If the statement is true, it will execute the block of code within the if statement. If it's false, and there's an else statement, it will execute the code within the else statement.

Example:

```
var budget = 5000;
if (budget > 4000) {
  console.log("Budget is within the limit");
} else {
  console.log("Budget is beyond the limit");
}
```

Comparison Operators: We use these operators to compare values. Here are some examples:

- `>`: Greater than
- `<`: Less than
- `>=`: Greater than or equal to
- `<=`: Less than or equal to
- `==`: Equal to
- `!=`: Not equal to

Logical Operators: We use these operators to make more complex logical expressions:

- `&&` - AND
- `||` - OR

- `!` - NOT

You can learn more about comparison and logical operators here

Exercises

Exercise 1

Write a function `validateBudget` that takes the `budget` as input and checks if it is within a limit of 5000. If it's more than 5000, it should return "Budget is beyond the limit" and if less it should return "Budget is within the limit."

Starter Code:

```
function validateBudget(budget){
  // Add your code here
}
```

Solution:

```
function validateBudget(budget){
  if (budget > 5000) {
    return("Budget is beyond the limit");
  } else {
    return("Budget is within the limit");
  }
}
```

Exercise 2

Write a function `validateEntries` that validates entries in an array. Each entry should be a positive integer and less than 5000. It should return an array of valid entries.

Starter Code:

```
function validateEntries(entries){
  // Add your code here
}
```

Solution:

```
function validateEntries(entries){
  var validEntries = [];
  for(var i = 0; i < entries.length; i++){
    if(entries[i] > 0 && entries[i] < 5000){
      validEntries.push(entries[i]);
```

```
    }
  }
  return(validEntries);
}
```

Exercise 3

Create a function `validateLimit` that takes `budget` and `limit` as arguments and use the AND (&&) operator to check if the budget is positive AND within limit.

Starter Code:

```
function validateLimit(budget, limit){
  // Add your code here
}
```

Solution:

```
function validateLimit(budget, limit){
  if(budget > 0 && budget < limit){
     return("Budget is within limit");
  } else {
     return("Budget is beyond limit");
  }
}
```

Test Cases

```
const {validateBudget, validateEntries, validateLimit} =
require("./OdessaBudget");

test("Check if budget is beyond limit", ()=>{
  expect(validateBudget(6000)).toBe('Budget is beyond the limit');
});

test("Check if budget is within limit", ()=>{
  expect(validateBudget(4000)).toBe('Budget is within the limit');
});

test("Check if entries are valid", ()=>{
  expect(validateEntries([6000, 4000, 200, -600])).toEqual([4000, 200]);
});

test("Check if budget is positive and within limit", ()=>{
  expect(validateLimit(4000, 5000)).toBe('Budget is within limit');
});
```

Closing Story

With a few lines of code, Odessa has made her work more efficient. Now, whenever there's a budget entry, the script is run and it returns a verdict - if the budget is within range or over limit. She's also managed to clean up the older entries to only include positive values and those less than 5000.

In the process, she's not only learned how to use if...else statements, comparison, and logical operators, but she also saved her time ⏰ for more important tasks ahead 💡. This was another step on her journey to becoming a full-stack developer, proving to her that with coding, every problem - no matter how daunting - can indeed have a solution.