MISSION READY

DARE TO
DEVELOP

Conditionals and Logical Operators

Reuben Simpson

Conditionals

- Conditionals in JavaScript are used to decide which block of code to run depending on some “condition”.
- There are three fundamental conditional statements that we can make use of in JavaScript:
  - `if`
  - `else`
  - `if else`
- “If” statements can be used on their own to run some code that passes a condition, but “else” and “if else” statements will always come after an “if” statement to run some other block of code if the initial condition is false.

If statements

- Conditionals syntax:

```
if (1 === 1) {
  console.log("The condition is true");
}
```

- First, we write the if statement, then we include our conditional statement between paratheses. In this case the statement is 1 === 1.
- The conditional statement should always evaluate to either true or false.
- If the condition is true, the code within the “if” statement is executed, otherwise the program continues.

If statements continued...

- We can also use variables in our conditional statement.

```
const x = 10;

if (x > 1) {
  console.log("x is greater than 1");
}
```

- Try it yourself, create your own if statement.

Else statements

- Conditionals syntax:

```
const x = 10;

if (x > 1) {
  console.log("x is greater than 1");
} else {
  console.log("x is less than 1");
}
```

- In this example we've added an "else" statement.
- If the condition next to the "if" statement is true then it will console.log that x is greater than 1, but if the condition is false then it will console.log that x is less than 1

If else statements

- You might've noticed something wrong with the last if statement

```
const x = 10;

if (x > 1) {
  console.log("x is greater than 1");
} else {
  console.log("x is less than 1");
}
```

- Changing the value of x to 1 will give us "x is less than 1" in the console log. But that isn't true. How can we fix that?
- We can add an if else statement.

If else statements continued…

- We can add another condition that will check to see if x is === to 1 and will console.log a different string.

```
const x = 1;

if (x > 1) {
  console.log("x is greater than 1");
} else if (x === 1) {
  console.log("x is 1");
} else {
  console.log("x is less than 1");
}
```

Exercise 2

- Write a JavaScript conditional statement that checks whether a variable is positive, negative or 0 and logs an appropriate message to the console.

Exercise 3

- If we take a string like `let myString = "1 apple";`
  - If we run `console.log(myString);`
  - It will give us `1 apple` in the console.
  - If I wanted to get just the first letter I could run `console.log(myString[0]);`
  - This would give me `1`
- Use this knowledge to create an if statement that will log to the console True if the string starts with an “a” and will log to the console False if it does not.

Ternary Operator

```
const word = "hippopotamus";

// The ternary operator
word.length > 10 ? console.log("The word is long") : console.log("The word is NOT long. It is short.");
```

- In this example we can see an alternative conditional statement called the ternary operator. This statement can be broken into 3 parts.
- The first part is the condition that will either be true or false (word.length > 10)
- Then we have the “?” section. Anything that comes after the question mark will execute if the condition is true.
- Lastly, we have the “:” section. Anything after the colon will execute if the condition is false.

Ternary vs Traditional

```
const word = "hippopotamus";

// Ternary operator
word.length > 10 ? console.log("The word is long") : console.log("The word is NOT long. It is short.");

// Is the same as writing:

// Traditional 'if else' statement
if (word.length > 10) {
  console.log("The word is long");
} else {
  console.log("The word is NOT long. It is short.");
}
```

- The reason for using the *ternary operator* over a traditional *if* statement is to keep things simple and concise (takes up only one line of code instead of 5).
- A general rule of thumb is that if there are only two possible outcomes e.g true or false, then a ternary is ok to use. But if there are multiple possible outcomes then using the traditional if else statement is preferred

Ternary exercise

- Convert the following if else statement into a ternary

```
const word = "hippopotamus";

if (word[0] === "h") {
  console.log("The word might be hippopotamus");
} else {
  console.log("The word is definitely not hippopotamus");
}
```

Logical operators

- Now that we have an understanding of what comparisons and conditionals are, we can take a look at some other types of operators.
- Let's run a check to see if a word is just long (10 – 15) chars or REALLY long (>15 chars)

```
const word = "responsiveness";

// first check if the word has more than 10 characters
if (word.length > 10) {
  // now check if the word has less than 15 characters
  if (word.length < 15) {
    console.log("The word is long");
  } else {
    console.log("The word is REALLY long");
  }
}
```

- This is a nested if statement (an if statement inside an if statement)
- It checks to see if one condition is true and if it is, then it will check if the second condition is true
- Let's make this shorter

Logical operators

- We can use something called a logical operator, we have two of them:
  - AND `&&`
  - OR `||`
- We can use the AND operator to check if more than one condition is true then run our code
  e.g. if word.length > 10 AND if word.length < 15
- We can use the OR operator to check if at least one condition is true
  e.g. if word.length === 0 OR if word.length === 10 then we run some code.

How to use logical operators

- The AND operator looks like “&&”
- The OR operator looks like “||”
- Let’s rewrite that example from before using the && operator

```
const word = "responsiveness";

if (word.length > 10 && word.length < 15) {
  console.log("The word is long");
} else if (word.length >= 15) {
  console.log("The word is REALLY long");
}
```

- Using an || operator could look like:

```
if (word.length === 0 || word.length > 25) {
  console.log("The string is empty or is greater than 25 characters!");
}
```

Switches

- Switches are a variation on conditionals that allow us to define many different blocks of code, each with their own conditions.
- If our switch statement matches one of the defined cases, then that block will execute, otherwise the default case (if one is defined) is executed.
- When comparing our given switch statement to one of the cases, switch uses the “strict” comparison i.e. === instead of the less strict ==

Switches continued…

Let’s take a look at the following example:

- We declared a variable “fruit” and set its value to "banana".
- We created a switch statement and passed into it the fruit variable.
- Switch will now compare all its cases with the provided fruit.
- If one of the cases match the condition we passed into the switch statement, (so if the fruit variable === the case fruit), then that case will be executed.
- Otherwise, the default case is executed.

```
let fruit = 'banana';

switch (fruit) {
    case 'orange':
        console.log('the fruit is an orange');
        break;
    case 'strawberry':
        console.log('the fruit is a strawberry');
        break;
    case 'banana':
        console.log('the fruit is a banana');
        break;
    default:
        console.log('fruit not found')
}
```

Review Questions:

- Where can we write JavaScript?
- What syntax & tag is used in an HTML file to add JS?
- What is the console.log()?
- What is a variable?
- How do you declare a variable?
- How do you assign a value to a variable?
- Explain: String, Numeric, Boolean

MISSION READY

www.missionreadyhq.com

DARE TO DEVELOP

Thank you

Reuben Simpson