

Lesson:

Ternary Conditions

Topics Covered :

1. Introduction to ternary operator.
2. Why majority of developers use the ternary operator?
3. Syntax of Ternary operator.
4. Code Implementation.
5. Chaining using the Ternary operator.

Operators are used to assigning, compare, and evaluate one, two, or more operands. We have different types of operators in JavaScript. These include comparison, arithmetic, logic, ternary, etc. Ternary operators have 3 blocks and are often used as a shorthand for an if-else statement.

The majority of developers use the ternary operator because:

1. The ternary operator allows you to write simple if-else statements in a single line of code, making your code more compact and readable.
2. The ternary operator can make your code more expressive by allowing you to clearly express the intent of the code in a single line.
3. The ternary operator is simple to use and understand, making it a good choice for short and simple conditions.
4. The ternary operator is widely used in several libraries and frameworks such as React.js where it's widely used.

Syntax of Ternary Operator:

Let's look at the syntax compared with the if-else statement.

```
// if-else statement
```

```
if (condition) {  
    expressionIfTrue;  
} else {  
    expressionIfFalse;  
}
```

The if-else statement takes a condition that will be evaluated to be either true or false.

The if-else statement above can be rewritten with ternary operators.

```
// Ternary Operator
```

```
condition ? <expressionIfTrue> : <expressionIfFalse>
```

Let's look at an example. Assume that you need to check if the person is logged in or not and provide the access to PW Skills lab.

We can do this in either of the ways:

1. Using if-else statements.
2. Using the ternary operator.

```

var isTheUserLoggedIn = true;

// Using if-else statement

if (isTheUserLoggedIn) {
  console.log("PW Skills lab Access Granted !!");
} else {
  console.log("PW Skills lab Access Denied !!");
}

var isTheUserLoggedIn = true;

// Using ternary operator

isTheUserLoggedIn
  ? console.log("PW Skills lab Access Granted !!")
  : console.log("PW Skills lab Access Denied !!");

```

While learning if-else statements we have seen how to chain if-else multiple statements.

In the same way, we can chain multiple ternary operators.

Let's assume in order to access the "Full Stack Web Developer Course", the user must be both logged in and should have purchased the course. Let's see how can we do this using the ternary operator.

```

var isTheUserLoggedIn = false;

var isTheCoursePurchased = false;

isTheUserLoggedIn
  ? isTheCoursePurchased
    ? console.log("Access Granted")
    : console.log("Access Denied!! Please Buy The Course")
  : console.log("Access Denied!! Please Login");

// OUTPUT : Access Denied!! Please Login

var isTheUserLoggedIn = true;

var isTheCoursePurchased = false;

isTheUserLoggedIn
  ? isTheCoursePurchased
    ? console.log("Access Granted")
    : console.log("Access Denied!! Please Buy The Course")
  : console.log("Access Denied!! Please Login");

// OUTPUT : Access Denied!! Please Buy The Course

```

```
var isTheUserLoggedIn = true;  
var isTheCoursePurchased = true;
```

```
isTheUserLoggedIn  
  ? isTheCoursePurchased  
    ? console.log("Access Granted !!")  
    : console.log("Access Denied!! Please Buy The Course")  
  : console.log("Access Denied!! Please Login");
```

```
// OUTPUT : Access Granted !!
```

Using chaining in ternary operators is not advised as they are not readable and debugging would be complex.