JavaScript if

Summary: in this tutorial, you will learn how to use the JavaScript `if` statement to execute a block when a condition is `true`.

Introduction to the JavaScript if statement

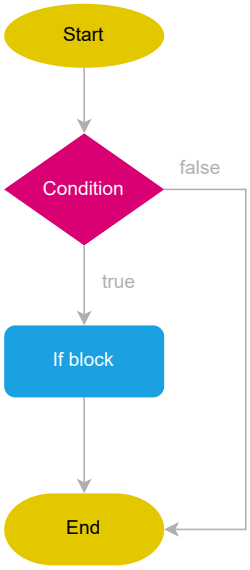
The `if` statement executes block if a condition is `true`. The following shows the syntax of the `if` statement:

```
if( condition )
   statement;
```

The `condition` can be a value or an expression. Typically, the condition evaluates to a boolean value, which is `true` or `false`.

If the `condition` evaluates to `true`, the `if` statement executes the `statement`. Otherwise, the `if` statement passes the control to the next statement after it.

The following flowchart illustrates how the `if` statement works:



If the `condition` is a non-boolean value, JavaScript will coerce it to a boolean value by calling the `Boolean()` function.

If you have more than one statement to execute, you need to wrap them in a block using a pair of curly braces (`{}`) as follows:

```
if (condition) {
  // statements to execute
}
```

It's a good practice to always use curly braces with the `if` statement. This makes your code easier to maintain and helps avoid possible mistakes.

JavaScript if statement examples

The following example uses the `if` statement to check if the age is equal to or greater than `18`:

```
let age = 18;
if (age >= 18) {
  console.log('You can sign up');
}
```

Output:

```
You can sign up
```

How it works.

First, declare and initialize the variable `age` to `18`:

```
let age = 18;
```

Second, check if the age is greater or equal to `18` using the `if` statement. Because the expression `age >= 18` is `true`, the code inside the `if` statement executes that outputs a message to the console:

```
if (age >= 18) {
  console.log('You can sign up');
}
```

The following example also uses the `if` statement. However, the `age` is `16` which causes the condition to be evaluated to `false`. Therefore, you won't see any message in the output:

```
let age = 16;
if (age >= 18) {
  console.log('You can sign up');
}
```

Nested if statement

It's possible to use an `if` statement inside another `if` statement. For example:

```
let age = 16;
let state = 'CA';
```

```
if (state == 'CA') {
  if (age >= 16) {
    console.log('You can drive.');
  }
}
```

Output:

```
You can drive.
```

How it works.

First, declare and initialize the `age` and `state` variables:

```
let age = 16;
let state = 'CA';
```

Second, check if the `state` is `'CA'` using an `if` statement. If yes, check if the `age` is greater than `16` using a nested `if` statement and output a message to the console:

```
if (state == 'CA') {
  if (age == 16) {
    console.log('You can drive.');
  }
}
```

In practice, you should avoid using nested `if` statements as much as possible.

For example, you can use the logical AND operator `&&` to combine the conditions and use an `if` statements as follows:

```
let age = 16;
let state = 'CA';

if (state == 'CA' && age == 16) {
  console.log('You can drive.');
}
```

Summary

- Use the JavaScript `if` statement to execute a statement if a condition is `true` .
- Avoid using nested `if` statement as much as possible.

Quiz