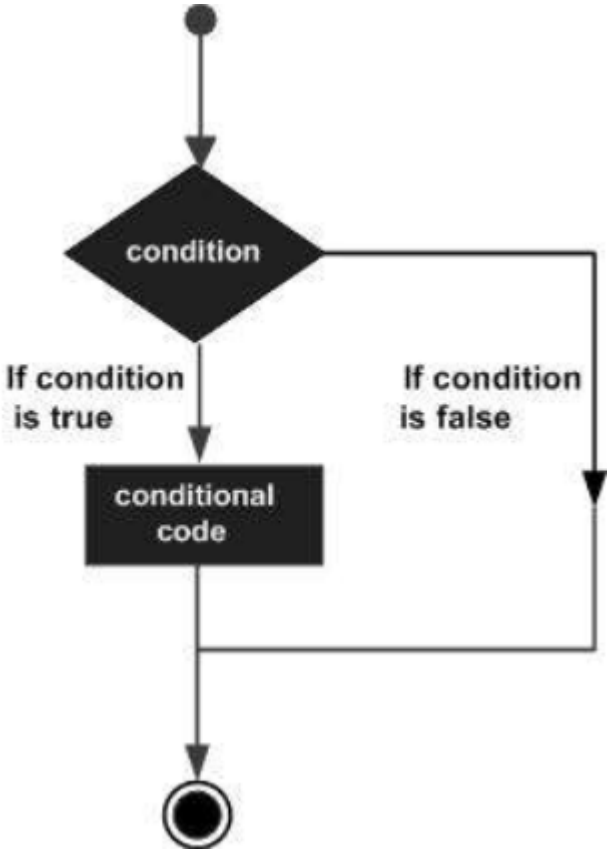
ANECO ACADEMY

JS-DECISION-ARRAY_METHODS

JavaScript conditional statements

JavaScript includes if-else conditional statements to control the program flow, similar to other programming languages.

- if condition
- if-else condition
- else if condition



Conditional statements - Else condition

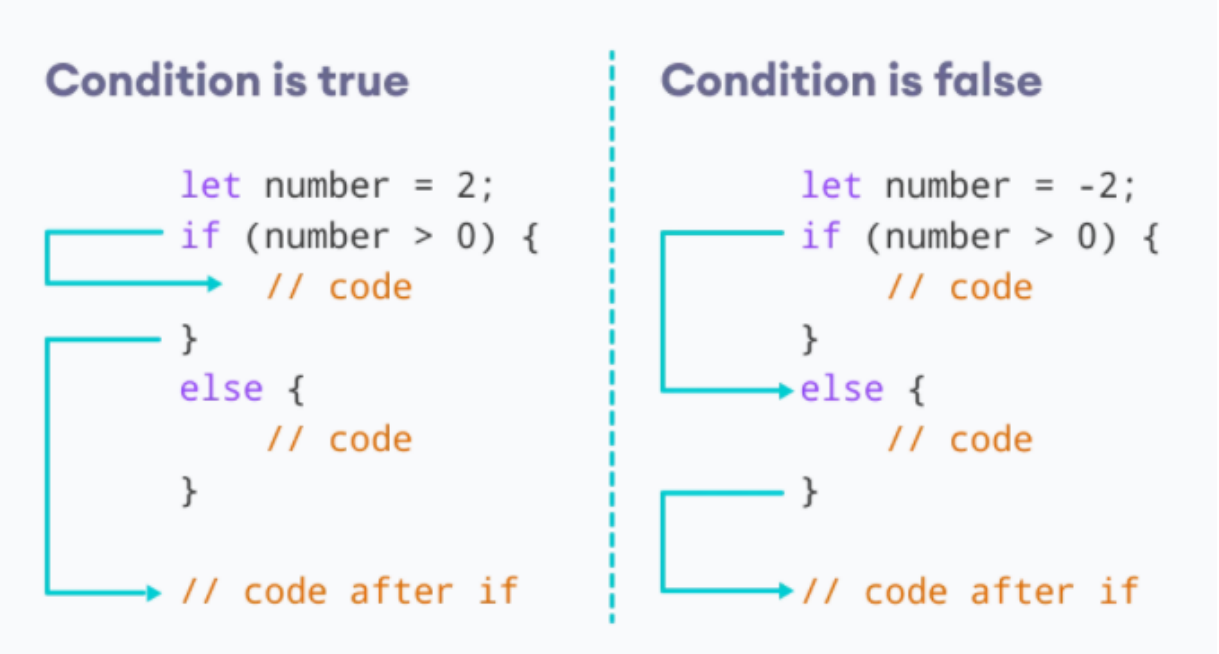
Use else statement when you want to execute the code every time when if condition evaluates to false.

Syntax:

```
if(condition expression)
{
   //  block of code to be executed if the condition is true}
else{
//  block of code to be executed if the condition is false
}
```

Example:

```
let num = 2
// let num = -2
if(number > 0)
{
   alert(num +" is greater than 0");
}
else
{
   alert(num +" is less than 0");
}
```



Conditional statements – if condition

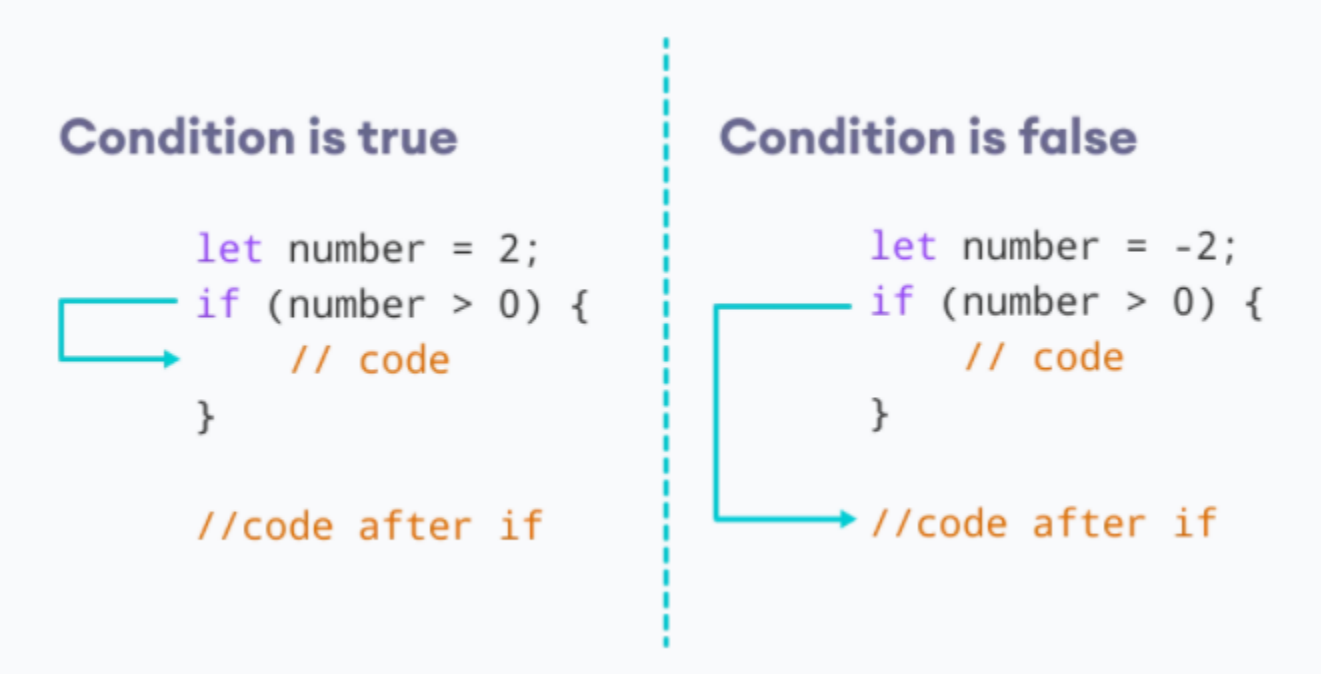
- The keyword if tells JavaScript to start the conditional statement.
- (1 > 0) is the condition to test, which in this case is true — 1 is greater than 0.
- The part contained inside curly braces {} is the block of code to run.
- Because the condition passes, the variable outcome is assigned the value "if block".

Syntax:

```
if(condition expression)
{
   // code to be executed if condition is true
}
```

Example:

```
if( 1 > 0)
{
   alert("1 is greater than 0");
}
if( 1 < 0)
{
   alert("1 is less than 0");
}
```



Conditional statements – Else if condition

Use "else if" condition when you want to apply second level condition after if statement

Syntax:

```
if(condition expression)
{
   //Execute this code block
}
else if(condition expression){
   //Execute this code block
}
```

Example:

```
// check if the number if positive, negative or zero
let number = 2;
number is positive");
} if (number > 0) {
   console.log("The
else if (number == 0) {
  console.log("The number is 0");
}
else {
   console.log("The number is negative");
}
```

1st Condition is true

```
let number = 2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

2nd Condition is true

```
let number = 0;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

All Conditions are false

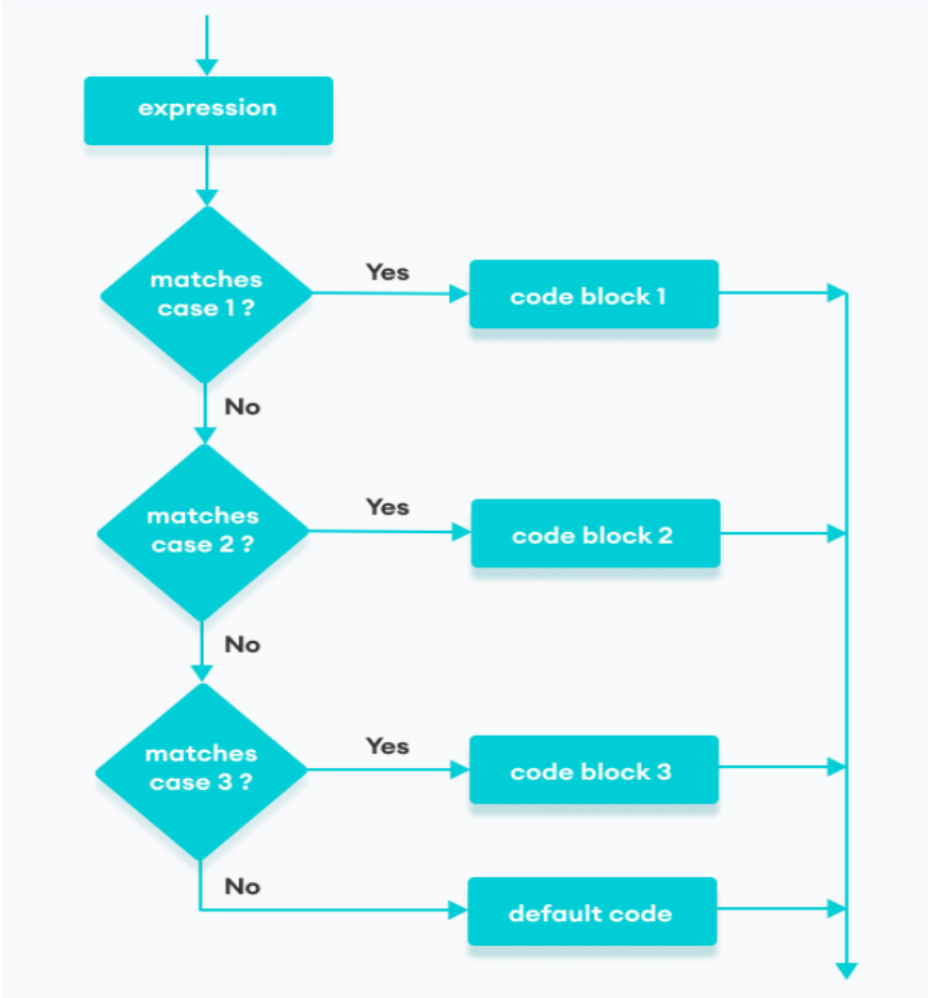
```
let number = -2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

JavaScript conditional statements - Switch

- The JavaScript switch statement is used in decision making.
- The switch statement evaluates an expression and executes the corresponding body that matches the expression's result.

Syntax:

```
switch(expression or literal value){
   case 1:
      //code to be executed
     break;
   case 2:
      //code to be executed
      break;
   case n:
      //code to be executed
      break;
   default:
      //default code to be executed
      //if none of the above case executed
}
```



JavaScript conditional statements - Switch

Example:

```
let a = 2;

switch (a) {

   case 1:
      a = 'one';
      break;
   case 2:
      a = 'two';
      break;
   default:
      a = 'not found';
      break;
}
console.log(`The value is ‘+{a});
```

Example:

```
let str = "bill";

switch (str)
{
   case "steve":
      alert("This is Steve");
   case "bill":
      alert("This is Bill");
      break;
   case "john":
      alert("This is John");
      break;
   default:
      alert("Unknown Person");
      break;
}
```

Array Methods

An array is an object that can store multiple values, create an array is by using an array literal []

Arrays to Strings:

```
const elec = [“Mobile", “Ipad", “Laptop", “Chargers"];
console.log(elec.toString())
```

join():

```
const elec = [“Mobile", “Ipad", “Laptop", “Chargers"];
console.log(elec.join("-"))
//mobile-ipad-laptop-charges
```

Add an Element to an Array:

push() and unshift() to add elements to an array

```
let dailyActivities = ['eat', 'sleep'];

// add an element at the end
dailyActivities.push('exercise');

console.log(dailyActivities); //  ['eat', 'sleep', 'exercise']
```

```
let dailyActivities = ['eat', 'sleep'];

//add an element at the start
dailyActivities.unshift('work');

console.log(dailyActivities); // ['work', 'eat', 'sleep']
```

Array Methods

Change the Elements of an Array:

Add elements or Change the elements by accessing the index value

```
let dailyActivities = [ 'eat', 'sleep'];

// this will add the new element 'exercise' at the 2 index
dailyActivities[2] = 'exercise';

console.log(dailyActivities); // ['eat', 'sleep', 'exercise']
```

Remove an Element from an Array:

pop() method to remove the last element from an array. The pop() method also returns the removed element

```
let dailyActivities = ['work', 'eat', 'sleep', 'exercise'];

// remove the last element
dailyActivities.pop();
console.log(dailyActivities); // ['work', 'eat', 'sleep']

// remove the last element from ['work', 'eat', 'sleep']
const removedElement = dailyActivities.pop();
```

shift() method removes the first element and also returns the removed element

```
let dailyActivities = ['work', 'eat', 'sleep'];

// remove the first element
dailyActivities.shift();

console.log(dailyActivities); // ['eat', 'sleep']
```

Array Methods

Array Length:

Number of elements in an array using the length property

```
const dailyActivities = [ 'eat', 'sleep'];

// this gives the total number of elements in an array
console.log(dailyActivities.length); // 2
```

Concat:

The concat() method creates a new array by merging existing array

```
const myDailyAct = [ 'eat', 'sleep'];
const myAddiction = [“work", “play", “roam"];

const myChildren = myDailyAct.concat(myAddiction);
```

splice():

- The splice() method can be used to add new items to an array
- The first parameter (2) defines the position **where** new elements should be **added**.
- The second parameter (0) defines **how many** elements should be **removed**.
- The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be **added**.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");

const remItems = fruits.splice(2, 2, "Lemon", "Kiwi");

// to remove items
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(0, 1);
```

Array Methods

slice():

- The slice() method slices out a piece of an array into a new array
- The slice() method does not remove any elements from the source array.

```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = fruits.slice(2);

//slice() method can take two arguments like slice(1, 3)
const citrus = fruits.slice(1,4);
```

Sorting an Array:

The sort() method sorts an array alphabetically

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();
```

Reversing an Array:

The reverse() method reverses the elements in an array

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.reverse();
```

Array Methods

indexOf():

searches an element of an array and returns its position

```
//finding the index position of string
const position = dailyActivities.indexOf('work');
console.log(position); // 2
```

includes():

The includes() method returns true if an array contains a specified value

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.includes("Mango");
```

isArray():

Check if an object is an array

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let result = Array.isArray(fruits); // true

let text = “testStringArray";
let result = Array.isArray(text); // false
```