

**If…else**:

In the last section, you learned about expressions and operators.

In this section we are going to use these expressions and operators along with conditional statements to implement interactivity in our applications.

In JavaScript we have two types of conditional statements.



Let us start with If…else, Its *Basic structure* or *Syntax*.

if (condition1) {

  statement1;

} else if (condition2) {

  statement2;

} else if (condition3) {

  statement3;

} else statement4;

Note: The space between curly braces is the block of code, this block of code can have single or multiple statements.

If the condition is set to true the relative block of code is executed.

**Example**:

If hour is between 6 to 12 : Good Morning

If hour is between 12 to 16: Good Afternoon

Otherwise good evening

Solution:

let hour = 19;

if (hour >= 6 && hour <= 12) {

  console.log("Good Morning");

} else if (hour >= 12 && hour <= 18) {

  console.log("Good Afternoon");

} else console.log("Good Night");

**Switch…Case**:

Let us start by declaring a variable called *role*, Imagine this represents role of the current user. Now we want to see whether this user is a guest or a moderator or an admin.

let role;

We can easily implement this logic using a bunch of if-else blocks, but there is another way using *switch-case*.

switch(variable) {

}

Instead of adding a condition after switch keyword, we use a variable.

switch(variable) {

  case 'anotherValue':

    console.log("variable = anotherValue")

}

Inside the curly braces, we add one or more case statements where each case statement is used to compare the value of variable with some another value.

Back to our example, we need to compare our role variable with some other roles.

switch (role) {

  case "guest":

    console.log("Guest User");

}

At the end of our case, *we need to add a break statement so that we jump out of the switch block, otherwise all the case statements inside switch block will be executed*.

let role = "guest";

switch (role) {

  case "guest":

    console.log("Guest User");

    break;

  case "moderator":

    console.log("Moderator User");

    break;

  default:

    console.log("Unknown user");

}

Note: If none of the cases match value then code inside *default* statement block will

With switch and case we can compare a variable against a bunch of other values. Now they do not have to be string, they can be numbers or even Booleans. *We can implement the same logic with if – else which is cleaner and shorter*.

**For** :

Sometimes you need to repeat an action a number of times. For example let us imagine we want to display hello world 5 times on the console. The best way is to use *loops*.

In JavaScript we have different kinds of loops and all these loops essentially do the same thing; *They repeat an action a number of times*. Below are their names.

🡨 All these loops essentially do the same thing but *there is a subtle difference between how they start and end*.

for (let i = 0; i < 5; i++) {

  console.log("Hello World", i);

}

Note: Here i is the *loop variable*, i < 5 is *condition* and i++ is *increment expression*.

Inside for we need three statements

🡪**initial expression** (*where we initialize a loop variable*)

🡪**condition** where we compare value of loop variable to some other value(*loop will run until the condition stays true*)

🡪**increment expression**.(*we use increment operator to increment value of i with one*)

Inside the curly braces or code block we can have one or multiple statements.

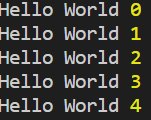
In the background value of i is incrementing as well with each iteration. Let us print value of i in console as well.

for (let i = 0; i < 5; i++) {

  console.log("Hello World", i);

}

O/P:



**While**:

One key difference between for and while loop is that *in for loops, the loop variable is part of the loop itself* *but in while loop we have to declare this variable externally*.

Print odd numbers between 1 to 5 on the console using while loop.

let number = 0;

while (number <= 5) {

  if (number % 2 != 0) {

    console.log(number);

  }

  number++;

}

Statements in while block will keep executing until the conditions are true.

**Do While**:

Very similar to while loop but they are slightly different. The statement inside do block, *execute at least once even if the condition is evaluated to false*.

Print odd numbers between 1 to 5 on the console using do-while loop.

let i = 0;

do {

  if (i % 2 != 0) {

    console.log(i);

  }

  i++;

} while (i <= 5);

Now if we change the initial expression value to 9, while statement will be false already.

let i = 9;

do {

  if (i % 2 != 0) {

    console.log(i);

  }

  i++;

} while (i <= 5);

O/P: 9

We still get the output once,

*This is because* ***while*** *condition is evaluated at the* ***end of do block*** *not at the beginning*.

**Infinite Loops**:

While writing loops , something you need to be aware of is an *infinite loop*.

let i = 0;

while (i < 5) {

  console.log(i);

  //i++;

}

If we do not increment value of i, the loop will run forever and our browser will crash.

*Infinite do while loop*:

let i = 0;

do{

  //i++;

}while(i<5);

**For…in**:

let person = {

  name: "Himanshu",

  age: 30,

};

Consider this person object having key – value pairs. If we want to iterate and console log its keys, we can use a ***for-in*** loop.

for (let key in person) {

  console.log(key); //name age

}

What if want to display value of each key as well. Earlier we learned that there are two ways to access property of an object.

*Using dot notation*,

person.name

*Using bracket notation*,

Person[‘name’]

Since the name of key is changing with every iteration, we will use bracket notation like this,

let person = {

  name: "Himanshu",

  age: 30,

};

for (let key in person) {

  console.log(key, person[key]);

}



So, This is the *for – in* loop that we use to iterate over properties of an object.

*for – in arrays*:

let colors = ["red", "green", "blue"];

for (let index in colors) {

  console.log(index); //0 1 2

}

In case of arrays , we get index of items in each iteration.

If you want to get the item at a given index,

for (let index in colors) {

  console.log(index, colors[index]);

}



Since the ES6 we have better way to iterate over arrays and that is our next topic.

**For…of**:

Instead of *in*, we use ***of*** keyword in the for loop.

let colors = ["red", "green", "blue"];

for (let color of colors) {

  console.log(color); //red green blue

}

With this type of for loop, we do not have to deal with index. We can directly deal with items of an array.

**Break and Continue**:

With all the loops we have learned about in this section, there are two keywords ***break*** and ***continue*** that can change how the loop behaves.

For this example, we will use a *while* loop. But it applies on all loops we have learned so far.

let i = 0;

while (i <= 10) {

  console.log(i); //0 1 2 3 4 5 6 7 8 9 10

  i++;

}

Let us add a *break* statement here,

let i = 0;

while (i <= 10) {

  if (i === 5) break;

  console.log(i); //0 1 2 3 4

  i++;

}

As we can see that as soon as value of i becomes 5, loop terminates.

Now let us look at *continue* keyword,

let i = 0;

while (i <= 10) {

  if (i % 2 === 0) {

    i++;

    continue;

  }

  console.log(i); // 1 3 5 7 9

  i++;

}

In the if condition we first check whether i is even or not. If true then we increment i and *when the JavaScript engine see the* ***continue*** *keyword, it will jump to beginning of loop*.

If condition not true then we console log the value if i and increment it by one.

So to recap,

***“****With* ***Break*** *keyword we jump out of a loop and with* ***continue*** *keyword we jump to next iteration****”***.

**Exercise 1**: Write a function that takes two numbers and returns the maximum of two.

*Using conditional statement*(*cleaner approach*):

function max(a, b) {

  return a > b ? a : b;

}

**Excercise2**: Landscape or Portrait

function isLandscape(width, height){

}

Implement this function which takes width and height of an image and returns True if image is Landscape(width > height) otherwise it returns false.

function isLandscape(width, height) {

  return width > height;

}

console.log(isLandscape(4000, 1000)); //true

**Excercise3**: FizzBuzz

Create a function called FizzBuzz and give it an input and it returns a string with some conditions.

If number is divisible by 3 🡪return Fizz

If number is divisible by 5 🡪 return Buzz

If number is divisible by 3 and 5 🡪 return FizzBuzz

If number is not divisible by 3 or 5 🡪 return Return the Number

If we do not pass a number 🡪 return “Not a Number”