Pakistan Blockchain Institute

MODULE-1

JAVASCRIPT CRASH COURSE

Class-5

Edversity.

Raja Rizwan Saleem
Lead Blockchain Trainer

Conditional Statements in JavaScript

Conditions

1. Up until now, all the code in our programs has been executed chronologically
2. Very often when you write code, you want to perform different actions for different decisions.
3. You can use conditional statements in your code to do this.

Conditions

1. In JavaScript we have the following conditional statements:
    a. Use ***if*** to specify a block of code to be executed, if a specified condition is true
    b. Use ***else*** to specify a block of code to be executed, if the same condition is false
    c. Use ***else if*** to specify a new condition to test, if the first condition is false
    d. Use ***switch*** to specify many alternative blocks of code to be executed

Conditions: if

1. The ***if*** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

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

}
```

Conditions: if

```
var age = 12;

if( age > 9 ) {
    console.log("Age = "+age);
}
```

Conditions: if

```
let temperature = 30;


if (temperature > 25) {
  console.log("It's hot outside!");
}
```

Conditions: if

```
let temperature = 30;


if (temperature > 25) {
  console.log("It's hot outside!");
}
```

Conditions: if

```
let age = 20;
let hasID = true;
let isMember = false;
if (age >= 18 && hasID || isMember) {
  console.log("Access granted.");
} else {
  console.log("Access denied.");
}
```

Conditions: else

1. Use the ***else*** statement to specify a block of code to be executed if the condition is false.

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

Conditions: else

```
var age =  15;


if( age > = 18 ) {
    console.log("Qualifies for driving");
} else {
    console.log("Does not qualify for driving");
}
```

Conditions: else if

1. Use the ***else if*** statement to specify a new condition if the first condition is false.

Conditions: else if

```
if (condition1) {
   //block of code to be executed if condition1 is true
} else if (condition2) {
   //block of code to be executed if the condition1 is
false  and condition2 is true
} else {
   //block of code to be executed if the condition1 is
false  and condition2 is false
}
```

Conditions: else if (practice)

```
var score = 80;
if( score > 80   ) {
    console.log("Grade A");
}  else if( score > 70 ) {  console.log("Grade B");
}  else if( score > 60 ) {  console.log("Grade C");
}  else {
    console.log("Failed");
}
```

Conditions: else if (practice)

```
let score = 85;
if (score >= 90) {
  console.log("Grade: A");
} else if (score >= 80) {
  console.log("Grade: B");
} else if (score >= 70) {
  console.log("Grade: C");
} else if (score >= 60) {
  console.log("Grade: D");
} else {
  console.log("Grade: F");
}
```

Real -World Examples of Conditional Statements

JavaScript Conditional Statements

Real-world Examples of Conditional Statements

Practical Applications of Conditional Statements in JavaScript

1

User Authentication and Authorization

Verify user identity and permissions before granting access to specific resources or functionalities.

2

Feature Toggles and Configurations

Control the availability of certain features or settings based on specified conditions or user roles.

3

Form Validation

Validate user input on web forms to ensure data accuracy and completeness before submission.

4

Dynamic Content Rendering

Render different content or user interfaces dynamically based on varying conditions or user interactions.

For Loop

For Loop

Loops

- A loop is a block of code that allows you to repeat a section of code a certain number of times, perhaps changing certain variable values each time the code is executed.

Loop

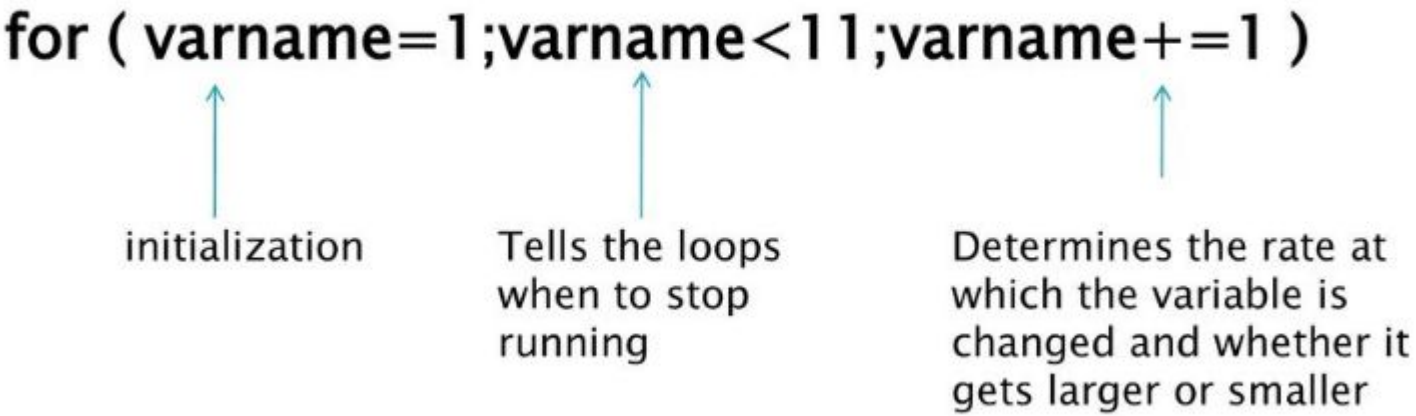
Why loops are useful?

- Loops are useful because they allow you to repeat lines of code without retyping them or using cut and paste in your text editor.
- They save time and trouble of repeatedly typing the same lines of code, but also avoids typing errors in repeated lines.
- You are also able to change one or more variable values each time the browser passes through the loop.

For Loop

1. Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true.
2. The for statement creates a loop that is executed as long as a condition is true.
3. It will only stop when the condition becomes false.

For Loop (structure)



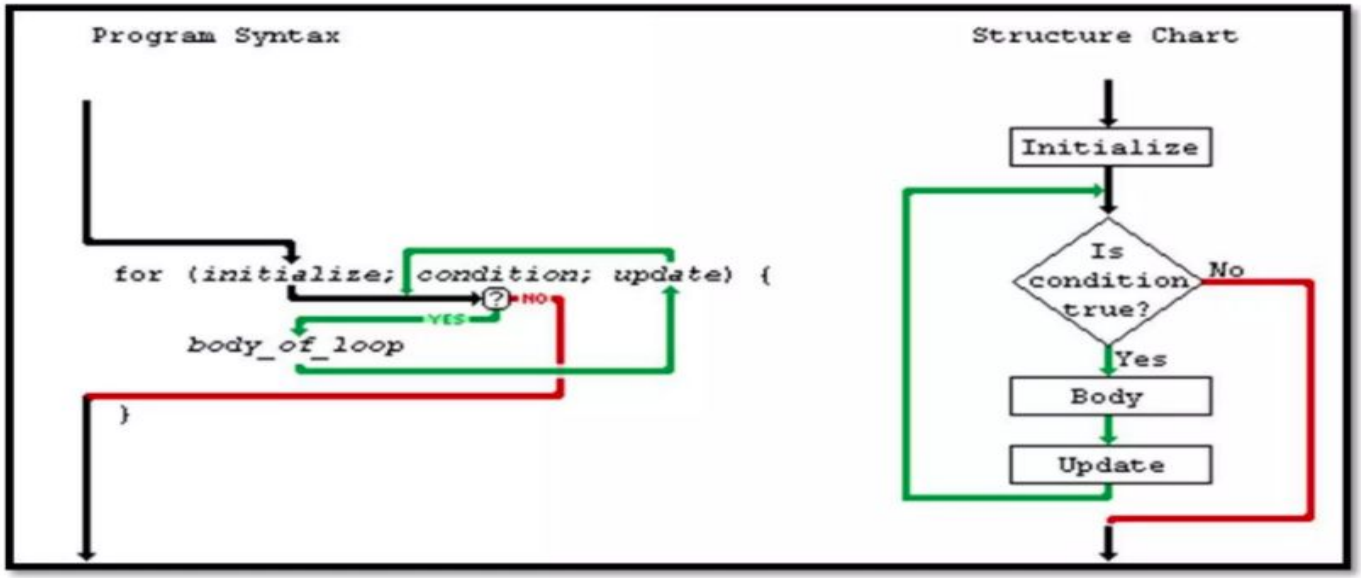
For Loop

```
for (initialization; condition; expression) {
    // code to be executed
}
```

1. Initialization is done (one time) before the execution of the code block.
2. Condition for executing the code block and exit loop
3. Expression is executed (every time) after the code block has been executed.

For Loop

The For Loop



For Loop

```
for (var i = 0; i < 10; i++){
    console.log(i);
}
```

Output: 0

1

2

For Loop

```
for (var i = 5; i <= 8;
    i++){  console.log(i);

}
```

Output:

```
5
6
7
8
```

For Loop (descending order)

```
for (var i = 6; i >=1; i--){
    console.log(i);
}
```

Output:

```
5
4
3
2
```

Infinite Loop

1. All 3 statement in loop are options, in that case it will be infinite loop
2. Also if you do not provide condition in loop it will make loop infinite

```
for ( ; ; ){
    console.log("Hello");
}
```

For Loop

1. Printing table of 3 will require hard coded statements

```
console.log("3 x 1 = 3");
console.log("3 x 2 = 6");
console.log("3 x 3 = 9");
console.log("3 x 4 = 12");
console.log("3 x 5 = 15");
```

For Loop to create tables

1. With for loop it will be dynamic

   Output:

   3 x 1 = 3

   ….

   3 x 10 = 30

While Loop

1.A "While" Loop is used to repeat a specific block of code an **unknown** number of times, until a condition is met.

For example, if we want to ask a user for a number between 1 and 10, we don't know how many times the user may enter a larger number, so we keep asking "while the number is not between 1 and 10". If we (or the computer) knows exactly how many times to execute a section of code (such as shuffling a deck of cards) we use a for loop.

While Loop

While loop

- A while loop just looks at a short comparison and repeats until the comparison is no longer true.

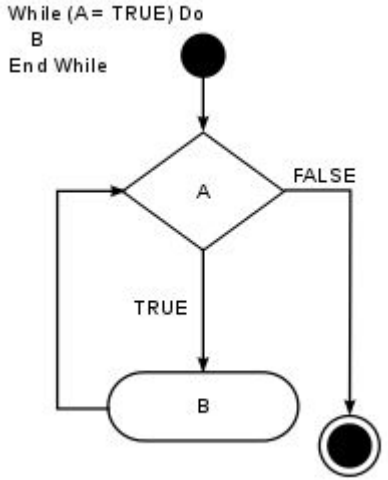
```
while(varname<11)
  {



  }
```

While Loop

```
let j = 0;
while (j < 10) {
  console.log(j+1);
  j +=1;
}
```



In the above example, the code in the loop will run, over and over again, as long as a variable (j) is less than 10

Do While Loop

```
let k =0;
    do {
        console.log(k+1);
         k +=1;
        }
while (k<10);
```

Break

```
for (var i = 0; i < 8;
    i++){   if(i == 4) {
        break;
    }
    console.log("I = "+i);
}
```

Output

: I = 0

I = 1

I = 2

I = 3

Continue

```
for (var i = 0; i < 8; i++){
    if(i == 4) {
        continue;
    }
    console.log("I = "+i);
}
```

Output

: I = 0

I = 1

I = 2

I = 3

I = 5

I = 6

I = 7

Nested Loops

1. If a loop exists inside the body of another loop, it's called nested loop.

```
for (var i = 0; i < 3; i++){
    for(var j = 0; j < 2; j++) {
        console.log("I = "+i+" J = "+j);
    }
}
```

Output:

I = 0 J = 0
I = 0 J = 1
I = 1 J = 0
I = 1 J = 1
I = 2 J = 0
I = 2 J = 1

Nested Loops

```
let a = ["a", "e", "i", "o", "u", "A", "E", "I", "O", "U"];
let input = prompt("please enter any character");
let check = false;
for (let i = 0; i < a.length; i++) {

if (input === a[i]) {

check = true;

console.log("It is a vowel"); break;
}
} if (check === false) {

console.log("it is a consonant"); }
```

Real -World Examples of Loops

JavaScript Loops Applications

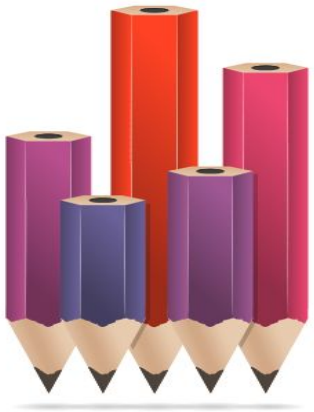
Real-world Examples of Loops

Exploring Practical Applications of Loops in JavaScript

Iterating over arrays and collections

Animation loops in web applications

Code snippets and live examples



Pagination and data fetching

Batch processing tasks

Thanks
End of Module-1 (Class-5)