**JavaScript (JS)** is the most popular lightweight, interpreted compiled programming language. It can be used for both Client-side as well as Server-side developments. It is also known as a scripting language for web pages. This free JavaScript Tutorial is designed to help both beginners and experienced professionals master the fundamentals of JavaScript and unleash their creativity to build powerful web applications.

**How to Execute** **JavaScript ?**

JavaScript can be executed inside one’s browser. You can open the JavaScript console and start writing JavaScript there.

Another way to execute JavaScript is a runtime like Node.js which can be used run JavaScript code.

**Expressions**

A fragment of code that produces a value is called an expression.

Some fundamental expressions in JavaScript:

**1. Arithmetic Expressions:**

* Addition: `a + b`
* Subtraction: `a - b`
* Multiplication: `a \* b`
* Division: `a / b`
* Modulus (Remainder): `a % b`
* Increment: `a++` or `++a`
* Decrement: `a--` or `--a`

**2. Comparison Expressions:**

* Equal to: `a == b`
* Not equal to: `a != b`
* Strict equal to: `a === b`
* Strict not equal to: `a !== b`
* Greater than: `a > b`
* Less than: `a < b`
* Greater than or equal to: `a >= b`
* Less than or equal to: `a <= b`

**3. Logical Expressions:**

* AND: `a && b`
* OR: `a || b`
* NOT: `!a`

4. Assignment Expressions:

* Assignment: `variable = value`
* Addition Assignment: `a += b`
* Subtraction Assignment: `a -= b`
* Multiplication Assignment: `a \*= b`
* Division Assignment: `a /= b`
* Modulus Assignment: `a %= b`

**Calculating the Area of a Rectangle:**

let length = 10;

let width = 5;

let area = length \* width;

console.log("Area of the rectangle:", area);

**Calculating the Hypotenuse of a Right Triangle:**

let base = 3;

let height = 4;

let hypotenuse = Math.sqrt(base\*\*2 + height\*\*2);

console.log("Hypotenuse of the right triangle:", hypotenuse);

**Calculating Simple Interest:**

let principal = 1000;

let rate = 0.05; // 5%

let time = 2; // in years

let simpleInterest = principal \* rate \* time;

console.log("Simple Interest:", simpleInterest);

**Converting Celsius to Fahrenheit:**

let celsiusTemperature = 25;

let fahrenheitTemperature = (celsiusTemperature \* 9/5) + 32;

console.log("Temperature in Fahrenheit:",fahrenheitTemperature);

**Calculating the Average of Three Numbers:**

let num1 = 15;

let num2 = 20;

let num3 = 25;

let average = (num1 + num2 + num3) / 3;

console.log("Average:", average);

**Comments**

In JavaScript, comments are used to add explanatory notes or annotations to your code. There are two types of comments in JavaScript:

**1. Single-line comments:**

- Single-line comments start with `//` and extend to the end of the line.

// This is a single-line comment

let x = 5; // Letiable assignment

**2. Multi-line comments:**

- Multi-line comments start with `/\*` and end with `\*/`. They can span multiple lines.

/\*

This is a multi-line comment

It can span multiple lines

\*/

let y = 10; /\* Another variable assignment \*/

Comments are ignored by the JavaScript interpreter and are not executed as code. They are useful for providing context, explaining code functionality, or temporarily disabling parts of the code for testing or debugging. It's good practice to include comments to make your code more readable and understandable, especially when working on larger projects or collaborating with others.

**Conditionals**

Conditional statements in JavaScript are used to control the flow of a program based on certain conditions. They allow you to execute different blocks of code depending on whether a specified condition evaluates to true or false.

**1. if Statement:**

- The `if` statement is used for basic conditional execution. If the specified condition evaluates to `true`, the block of code inside the `if` statement is executed.

if (condition) {

// Code to execute if the condition is true

}

**2. if-else Statement:**

- The `if-else` statement allows you to provide an alternative block of code to execute if the condition in the `if` statement is `false`.

if (condition) {

// Code to execute if the condition is true

} else {

// Code to execute if the condition is false

}

**3. if-else if-else Statement:**

- Use `else if` statements when there are multiple conditions to check. The first `if` or `else if` condition that evaluates to `true` will execute its block of code.

if (condition1) {

// Code to execute if condition1 is true

} else if (condition2) {

// Code to execute if condition2 is true

} else {

// Code to execute if none of the conditions are true

}

**4. Switch Statement:**

- The `switch` statement is used when there are multiple possible conditions to check against a single value.

switch (value) {

case condition1:

// Code to execute if value matches condition1

break;

case condition2:

// Code to execute if value matches condition2

break;

default:

// Code to execute if none of the cases match

}

**5. Ternary (Conditional) Operator:**

- The ternary operator (`? :`) provides a concise way to write simple conditional statements.

var result = (condition) ? trueExpression : falseExpression;

Conditional statements are essential for creating dynamic and responsive programs. They allow you to make decisions based on different conditions, enabling your code to adapt to various scenarios.

1. **Write a code to find out if someone is an Adult or a Minor.**

let age = 20;

if (age >= 18) {

console.log("You are an adult.");

} else {

console.log("You are a minor.");

}

1. **Write a code to greet according to time of the day.**

let hour = new Date().getHours();

let greeting;

if (hour < 12) {

greeting = "Good morning!";

} else if (hour < 18) {

greeting = "Good afternoon!";

} else {

greeting = "Good evening!";

}

console.log(greeting);

1. **Write a code to find if you will need an umbrella or not.**

let isRaining = true;

let weatherMessage = isRaining ? "Bring an umbrella!" : "No need for an umbrella.";

console.log(weatherMessage);

1. **Write a code to grade the students according to their marks.**

let score = 85;

let grade;

if (score >= 90) {

grade = "A";

} else {

if (score >= 80) {

grade = "B";

} else {

grade = "C";

}

}

console.log("Your grade is " + grade);

1. **Write a code to check which day of the week it is.**

let dayOfWeek = new Date().getDay();

let day;

switch (dayOfWeek) {

case 0:

day = "Sunday";

break;

case 1:

day = "Monday";

break;

case 2:

day = "Tuesday";

break;

case 3:

day = "Wednesday";

break;

case 4:

day = "Thursday";

break;

case 5:

day = "Friday";

break;

case 6:

day = "Saturday";

break;

default:

day = "Unknown";

}

console.log("Today is " + day);

**Loops in JavaScript**

Loops are essential constructs in programming that allow the repeated execution of a block of code. JavaScript provides several types of loops, each serving different purposes. Let's explore the main types of loops in JavaScript:

**1. `for` Loop:**

The `for` loop is commonly used when the number of iterations is known in advance.

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

// Code to be executed in each iteration

console.log(i);

}

In this example, the loop initializes a counter (`i`), checks a condition (`i < 5`), executes the code block, and increments the counter after each iteration.

**2. `while` Loop:**

The `while` loop continues to execute a block of code as long as the specified condition is true.

let count = 0;

while (count < 5) {

// Code to be executed in each iteration

console.log(count);

count++;

}

Here, the loop runs as long as `count` is less than 5.

**3. `do...while` Loop:**

Similar to the `while` loop, but it always executes the block of code at least once before checking the condition.

let num = 0;

do {

// Code to be executed in each iteration

console.log(num);

num++;

} while (num < 5);

Understanding and mastering loops is fundamental in JavaScript programming. They provide a powerful way to automate repetitive tasks and process data efficiently.

**Print numbers from 1 to 5**

for (let i = 1; i <= 5; i++) {

console.log(i);

}

**Print even numbers from 2 to 8**

let num = 2;

while (num <= 8) {

console.log(num);

num += 2;

}

**Print numbers from 1 to 5 using do...while**

let count = 1;

do {

console.log(count);

count++;

} while (count <= 5);

**Multiplication table up to 5x5**

for (let i = 1; i <= 5; i++) {

for (let j = 1; j <= 5; j++) {

console.log(i + ' \* ' + j + ' = ' + (i \* j));

}

}

**Generate random numbers until a certain condition is met**

let randomNumber;

do {

randomNumber = Math.random();

console.log(randomNumber);

} while (randomNumber < 0.8);

**Print odd numbers up to 10 using a while loop**

let num = 1;

while (true) {

console.log(num);

num += 2;

if (num > 10) {

break;

}

}

**String methods in JavaScript**

**1**. **Length:**

length: Returns the length of the string.

let myString = 'Hello, World!';

console.log(myString.length); // Outputs: 13

**2. Concatenation:**

**concat(): Joins two or more strings.**

let myString = 'Hello, World!';

let anotherString = " JavaScript is awesome!";

let combinedString = myString.concat(' ', anotherString);

console.log(combinedString); // Outputs: Hello, World! JavaScript is awesome!

**3. Substring:**

substring(startIndex, endIndex): Extracts characters between the specified indices.

let myString = 'Hello, World!';

let subset = myString.substring(0, 5);

console.log(subset); // Outputs: Hello

**4. Slice:**

slice(startIndex, endIndex): Extracts a section of a string.

let myString = 'Hello, World!';

let slicedString = myString.slice(0, 5);

console.log(slicedString); // Outputs: Hello

**5. IndexOf:**

- indexOf(searchString): Returns the index of the first occurrence of the specified substring.

let myString = 'Hello, World!';

let indexOfWorld = myString.indexOf('World');

console.log(indexOfWorld); // Outputs: 7

**6. LastIndexOf:**

- lastIndexOf(searchString): Returns the index of the last occurrence of the specified substring.

let myString = 'Hello, World!';

let lastIndexOfl = myString.lastIndexOf('l');

console.log(lastIndexOfl); // Outputs: 10

**7. charAt:**

charAt(index): Returns the character at the specified index.

let myString = 'Hello, World!';

let charAtIndex = myString.charAt(7);

console.log(charAtIndex); // Outputs: W

**8. Replace:**

replace(oldValue, newValue): Replaces a specified value with another value in a string.

let myString = 'Hello, World!';

let replacedString = myString.replace('World', 'Universe');

console.log(replacedString); // Outputs: Hello, Universe!

**9. toUpperCase / toLowerCase:**

toUpperCase(): Converts the string to uppercase.

toLowerCase(): Converts the string to lowercase.

let myString = 'Hello, World!';

console.log(myString.toUpperCase()); // Outputs: HELLO, WORLD!

console.log(anotherString.toLowerCase()); // Outputs: javascript is awesome!

**What is Array in JavaScript?**

JavaScript Array is a data structure that allows you to store and organize multiple values within a single variable. It is a versatile and dynamic object. It can hold various data types, including numbers, strings, objects, and even other arrays. Arrays in JavaScript are zero-indexed i.e. the first element is accessed with an index 0, the second element with an index of 1, and so forth.

**Basic Terminologies of JavaScript Array**

**Array**: A data structure in JavaScript that allows you to store multiple values in a single variable.

**Array Element**: Each value within an array is called an element. Elements are accessed by their index.

**Array Index**: A numeric representation that indicates the position of an element in the array. JavaScript arrays are zero-indexed, meaning the first element is at index 0.

**Array Length**: The number of elements in an array. It can be retrieved using the length property.

**Declaration of an Array**

There are basically two ways to declare an array i.e. Array Literal and Array Constructor.

**1. Creating an Array using Array Literal**

let arrayName = [value1, value2, ...];

**2. Creating an Array using Array Constructor (JavaScript new Keyword)**

let arrayName = new Array();

**Basic Operations on JavaScript Arrays**

**1. Accessing Elements of an Array**

Any element in the array can be accessed using the index number. The index in the arrays starts with 0.

let courses = ["HTML", "CSS", "Javascript", "React"];

console.log(courses[0]);

console.log(courses[1]);

console.log(courses[2]);

console.log(courses[3]);

**2. Accessing the First Element of an Array**

The array indexing starts from 0, so we can access first element of array using the index number.

let courses = ["HTML", "CSS", "JavaScript", "React"];

let firstItem = courses[0];

console.log("First Item: ", firstItem);

**3. Accessing the Last Element of an Array**

We can access the last array element using [array.length – 1] index number.

let courses = ["HTML", "CSS", "JavaScript", "React"];

let lastItem = courses[courses.length - 1];

console.log("First Item: ", lastItem);

**4. Modifying the Array Elements**

Elements in an array can be modified by assigning a new value to their corresponding index.

let courses = ["HTML", "CSS", "Javascript", "React"];

console.log(courses);

courses[1]= "Bootstrap";

console.log(courses);

**5. Adding Elements to the Array**

Elements can be added to the array using methods like push() and unshift().

let courses = ["HTML", "CSS", "Javascript", "React"];

// Add Element to the end of Array

courses.push("Node.js");

// Add Element to the beginning

courses.unshift("Web Development");

console.log(courses);

**6. Removing Elements from an Array**

Remove elements using methods like pop(), shift(), or splice().

let courses = ["HTML", "CSS", "Javascript", "React", "Node.js"];

console.log("Original Array: " + courses);

// Removes and returns the last element

let lastElement = courses.pop();

console.log("After Removed the last elements: " + courses);

// Removes and returns the first element

let firstElement = courses.shift();

console.log("After Removed the First elements: " + courses);

// Removes 2 elements starting from index 1

courses.splice(1, 2);

console.log("After Removed 2 elements starting from index 1: " + courses);

**7. Iterating Through Array Elements**

We can iterate array and access array elements using for and forEach loop.

Example: It is the example of for loop.

let courses = ["HTML", "CSS", "JavaScript", "React"];

// Iterating through for loop

for (let i = 0; i < courses.length; i++) {

console.log(courses[i])

}

Example: It is the example of Array.forEach() loop.

let courses = ["HTML", "CSS", "JavaScript", "React"];

// Iterating through forEach loop

courses.forEach(function myfunc(elements) {

console.log(elements);

});

**8. Array Concatenation**

Combine two or more arrays using the concat() method. Ir returns new array conaining joined arrays elements.

let courses = ["HTML", "CSS", "JavaScript", "React"];

let otherCourses = ["Node.js", "Expess.js"];

// Concatenate both arrays

let concateArray = courses.concat(otherCourses);

console.log("Concatenated Array: ", concateArray);

**9. Conversion of an Array to String**

We have a builtin method toString() to converts an array to a string.

let courses = ["HTML", "CSS", "JavaScript", "React"];

// Convert array ot String

console.log(courses.toString());

**JavaScript Miscellaneous Code**

**reverse a number**

**let originalNumber = 12345;**

**let reversedNumber = 0;**

**while (originalNumber !== 0) {**

**// Get the last digit of the original number**

**let digit = originalNumber % 10;**

**// Add the digit to the reversedNumber (shift digits to the left)**

**reversedNumber = reversedNumber \* 10 + digit;**

**// Remove the last digit from the original number**

**originalNumber = Math.floor(originalNumber / 10);**

**}**

**console.log(`Original Number: ${originalNumber}`);**

**console.log(`Reversed Number: ${reversedNumber}`);**