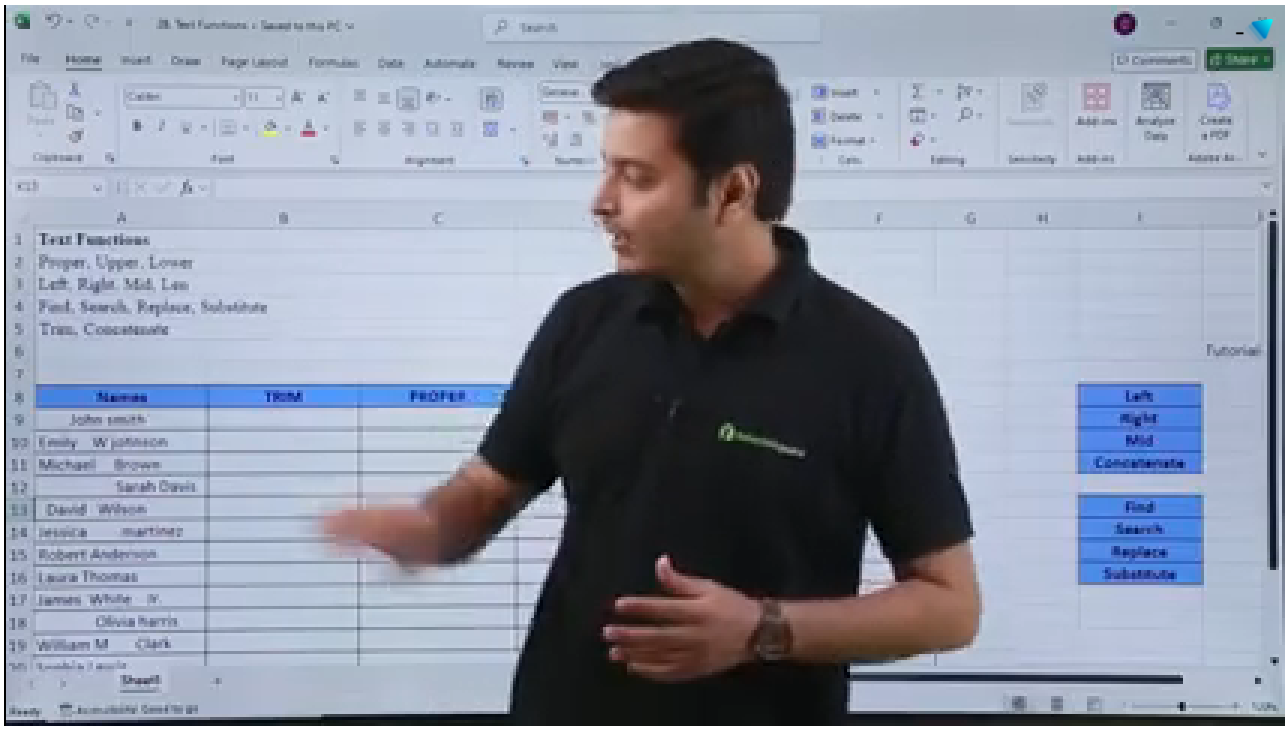
JavaScript Cheatsheet

Write On-page JavaScript

To add on-page JavaScript in an HTML document, use the following script tag −

```
<script type="text/javascript">
// Write your JavaScript code here
</script>
```

Insert an External JavaScript file

You can insert an external JavaScript file in an HTML document using the HTML's <script> tag by specifying the file path with the **src** attribute.

```
<script src="file.js"></script>
```

Printing/Output

JavaScript provides three ways to print anything on the screen.

- Console Output
- Alert Boxes
- Document Write

Console Output (console.log())

The console.log() prints on the web console. It is used for debugging and logging purposes.

</> Open Compiler

```
console.log("Hello, World!");
```

Alert Boxes

JavaScript alert() method displays a pop-up alert box on the browser with a specified message such as text, values of the variables, etc.,

</> Open Compiler

```
alert("Hello World!");
var x = 10;
alert("Value of x is :" + x);
```

document.write()

The document.write() method to write content directly to the HTML document.

</> Open Compiler

```
document.write("Hello World!");
var x = 10;
document.write("Value of x is :", x);
```

Variables

JavaScript variables can be declared using the **var**, **let**, or **const** keywords.

- **var** – Declares function-scoped variables and these can be reassigned.
- **let** – Declares block-scoped variables and these can be reassigned.
- **const** – Declares constants and their values cannot be reassigned.

```
var x = 5;
let y = 10;
const z = 15;
```

Data Types

JavaScript data types can be categorized into the following two categories –

1. Primitive Types

The primitive data types are: **String**, **Number**, **Boolean**, **Undefined**, **Null**, **Symbol**, **BigInt**

2. Objects Types

The objects data types are: **{}**, **arrays []**, **functions () => {}**

```
let str = "Kelly Hu";
let num = 123;
let bool = true;
let und = undefined;
let n = null;
```

Operators

The following are the JavaScript operators –

Arithmetic Operators

The arithmetic operators are: +, -, *, /, %, ++, --

</> Open Compiler

```
let a = 10;
let b = 3;
console.log("a =", a, ", b =", b);

// Addition (+)
let sum = a + b;
console.log("a + b =", sum);
// Subtraction (-)
let difference = a - b;
console.log("a - b =", difference);
// Multiplication (*)
let product = a * b;
console.log("a * b =", product);
// Division (/)
let quotient = a / b;
console.log("a / b =", quotient);
// Modulus (remainder) (%)
let remainder = a % b;
console.log("a % b =", remainder);
// Increment (++)
a++;
console.log("After a++:", a);
// Decrement (--)
b--;
console.log("After b--:", b);
```

Assignment Operators

The assignment operators are: =, +=, -=, *=, /=

</> Open Compiler

```
let x = 10;
console.log("x:", x);
```

```
x = 5;
console.log("x:", x);
x += 3;
console.log("x:", x);
x -= 2;
console.log("x:", x);
x *= 4;
console.log("x:", x);
x /= 6;
console.log("x:", x);
x %= 3;
console.log("x:", x);
```

Comparison Operators

The comparison operators are: ==, ===, !=, !==, >, =,

</> Open Compiler

```
let x = 5;
let y = "5";
let z = 10;

console.log(x == y);
console.log(x === y);
console.log(x != z);
console.log(x !== y);
console.log(z > x);
console.log(x < z);
console.log(z >= 10);
console.log(x <= 5);
```

Logical Operators

The logical operators are: **&&** (AND), **||** (OR), and **!** (NOT)

</> Open Compiler

```
let a = true;
let b = false;
let c = 5;
let d = 10;

console.log(a && c < d);
console.log(b && c < d);
console.log(a || b);
console.log(b || d < c);
console.log(!a);
console.log(!b);
```

Conditional Statements

JavaScript conditional statements contain different types of if-else statements and ternary operators.

If else statements

The syntax of if-else statements are –

```
if (condition) {
   // block of code
} else if (condition) {
   // block of code
} else {
   // block of code
}
```

Below is an example of if-else statements –

</> Open Compiler

```
let age = 20;

if (age < 18) {
   console.log("You are a minor.");
} else if (age >= 18 && age < 65) {
   console.log("You are an adult.");
} else {
```

```
   console.log("You are a senior citizen.");
}
```

Ternary Operator

The ternary operator is a replacement for a simple if else statement. Below is the syntax of the ternary operator –

```
let result = condition ? 'value1' : 'value2';
```

An example of a ternary operator is as follows –

</> Open Compiler

```
let age = 20;

let message = age < 18
   ? "You are a minor."
   : age >= 18 && age < 65
   ? "You are an adult."
   : "You are a senior citizen.";

console.log(message);
```

Loops

JavaScript loops are –

- The for Loop
- The while Loop
- The do-while Loop

The for Loop

The for loop is an entry control loop.

</> Open Compiler

```
for (let i = 0; i < 5; i++) {
   console.log(i);
}
```

The while Loop

The while loop is also an entry control loop where the condition is checked before executing the loop's body.

</> Open Compiler

```
let i = 0;
while (i < 5) {
   console.log(i);
   i++;
}
```

The do-while loop is an exit control loop, where the condition is checked after executing the loop's body.

</> Open Compiler

```
let i=0;
do {
   console.log(i);
   i++;
} while (i < 5);
```

Functions

JavaScript functions are the code blocks to perform specific tasks.

User-defined Function

Here is an example of a function to add two numbers –

</> Open Compiler

```
// Function Declaration
function addNumbers(a, b) {
   return a + b;  // Return the sum of a and b
}

// Example usage
let sum = addNumbers(5, 10);
console.log("The sum is:", sum);  // The sum is: 15
```

Function Expression

The function expression is as follows –

```
const multiply = function(a, b) {
   return a * b;
}
```

Arrow Function

JavaScript arrow function is used to write function expressions.

Below is a simple statement to create an arrow function –

```
const divide = (a, b) => a / b;
```

Example of an arrow function to add two numbers –

</> Open Compiler

```
// Arrow function
const addNumbers = (a, b) => a + b;
// Calling
let sum = addNumbers(5, 10);
console.log("The sum is:", sum);
```

Objects

JavaScript objects are collections of key-value pairs and are used to store different types of data, including other objects, functions, arrays, and primitive values.

Here is an example to create an object in JS –

```
const person = {
   name: "Kelly Hu",
   age: 27,
   display: function() {
      console.log("Hello, " + this.name);
   }
};

console.log(person.name);  // Access property
person.display();            // Call method
```

Arrays

JavaScript arrays store multiple values of any type in a single variable.

Array Declaration

Syntax to declare an array is –

```
let array_name = [value1, value2, value3, ];
```

Array Example

The following is an example of creating an array of integers.

```
let arr = [10, 20, 30, 40, 50];
// Printing array
console.log("arr:", arr);
```

Array Methods

Commonly used array methods are –

- **push()** – It is used to add one or more elements in an array.
- **pop()** – It is used to remove the last element and returns the deleted element.
- **shift()** – It is used to remove the first element and return the delete element.
- **unshift()** – It is used to add one or more elements at the beginning of an array.
- **concat()** – It is used to add one or more arrays and returns a new array.
- **join()** – It is used to join all elements of an array into a string.

Below is an example demonstrating all the above methods –

</> Open Compiler

```
// Initialize the original array
let students = ["Kelly Hu", "Peter", "John"];
// push()
students.push("Bobby", "Catty");
console.log(students);
// pop()
console.log("Removed :", students.pop());
// shift()
console.log("Removed :", students.shift());
// unshift()
students.unshift("Julia", "Halle");
console.log(students);
// concat()
const newNames = ["Emma", "Reese"];
const newArray = students.concat(newNames);
console.log("After concat:", newArray);
// join()
const str = students.join(", ");
console.log("Joined string:", str);
```

Loop through Array Elements

You can loop through all array elements using the forEach() method –

</> Open Compiler

```
var arr = [10, 20, 30, 40, 50]
arr.forEach((item) => console.log(item));
```

DOM Manipulation

JavaScript DOM manipulation allows you to manipulate the content and structure of web pages dynamically.

```
let element = document.getElementById('myElement');
element.innerHTML = 'New Content';  // Change content
element.style.color = 'red';        // Change style
document.querySelector('.class');   // Select by class
```

Event Listeners

JavaScript event listeners are allowed to execute code in response to various user actions, such as clicks, key presses, mouse movements, and more.

Below is an example of button click event –

```
element.addEventListener('click', function() {
  alert('Clicked!');
});
```

Promises

JavaScript promises represent the values that may be available now, or in the future, or never.

</> Open Compiler

```
// Promises
let promise = new Promise((resolve, reject) => {
   // asynchronous code
   if (success) resolve('Success');
   else reject('Error');
});
```

```
promise.then(result => console.log(result)).catch(err => console.log(err));
```

Async/Await

JavaScript Async/Await works with asynchronous operations.

The following is an example of Async/Await –

</> Open Compiler

```
async function fetchData() {
   try {
      let response = await fetch('url');
      let data = await response.json();
      console.log(data);
   } catch (error) {
      console.log(error);
   }
}
```

Error Handling

JavaScript error handling allows you to handle errors/exceptions that occur during runtime. The try, catch, and finally blocks are used to handle exceptions.

Syntax of error handling is –

```
try {
   // Code that may throw an error
} catch (error) {
   console.log(error.message);  // Handle the error
} finally {
   console.log("Finally block executed");
}
```

The following is a simple example demonstrating the use of try, catch, and finally in JavaScript –

</> Open Compiler

```
function divideNumbers(num1, num2) {
   try {
      if (num2 === 0) {
         throw new Error("Cannot divide by zero!");
      }
      const result = num1 / num2;
      console.log(`Result: ${result}`);
   } catch (error) {
      console.error("Error:", error.message);
   } finally {
      console.log("Execution completed.");
   }
}

// Calling
divideNumbers(10, 2);
divideNumbers(10, 0);
```

Ethical Hacking Tutorial

Docker Tutorial

Kubernetes Tutorial

DSA Tutorial

Spring Boot Tutorial

SDLC Tutorial

Unix Tutorial

CERTIFICATIONS

Business Analytics Certification

Java & Spring Boot Advanced Certification

Data Science Advanced Certification

Cloud Computing And DevOps

Advanced Certification In Business Analytics

Artificial Intelligence And Machine Learning

DevOps Certification

Game Development Certification

Front-End Developer Certification

AWS Certification Training

Python Programming Certification

COMPILERS & EDITORS

Online Java Compiler

Online Python Compiler

Online Go Compiler

Online C Compiler

Online C++ Compiler

Online C# Compiler

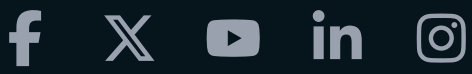
Online PHP Compiler

Online MATLAB Compiler

Online Bash Compiler

Online SQL Compiler

Online Html Editor



Tutorials Point is a leading Ed Tech company striving to provide the best learning material on technical and non-technical subjects.