
# SNS COLLEGE OF ENGINEERING


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# Javascript OBJECTS


# JAVASCRIPT OBJECTS


- In JavaScript, an **object** is a collection of properties, where each property is defined as a **key-value pair**. Objects allow us to store, organize, and manipulate data efficiently.
- Objects can contain **values**, including numbers, strings, arrays, functions, and even other objects.


# OBJECT PROPERTIES


- A real life car has **properties** like weight and color:
- car.name = Fiat, car.model = 500, car.weight = 850kg, car.color = white.
- Car objects have the same **properties**, but the **values** differ from car to car.
- Objects are variables too. But objects can contain many values.
- const car = {type:"Fiat", model:"500", color:"white"};


# OBJECT METHODS


- It is a common practice to declare objects with the const keyword.
- A real life car has **methods** like start and stop:
- car.start(), car.drive(), car.brake(), car.stop().
- Car objects have the same **methods**, but the methods are performed **at different times**.


# Using Object Literal


- The easiest way to create an object is by using curly braces {} and defining properties inside.

```
javascript
let person = { name: "Alice", age: 25, c
ountry: "India" }; console.log(person.name);
// Output: Alice
```


# Using the new Object() Constructor



- ```
  let student = new Object();
  student.name = "Bob";
  student.age = 22;
  console.log(student.name);
  ```
  ```


# Using the new Object() Constructor



- ```
  let student = new Object();
  student.name = "Bob";
  student.age = 22;
  console.log(student.name);
  ```
  ```


# Using the new Object() Constructor


- Create an Object

```
const person = {};

// Add Properties
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

# Using the new Object() Constructor

```
<p id="demo"></p>
<script>
// Create an empty Object
const person = {};
// Add Properties
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
// Display Data from Object
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
```


# SNS COLLEGE OF ENGINEERING


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## JAVASCRIPT DEBUGGERS


# DEBUGGERS


Debugging is an essential part of JavaScript development to find and fix errors in the code. A **debugger** is a tool that helps developers inspect and troubleshoot JavaScript code by pausing execution, stepping through the code, and checking variable values.


# Common Debugging Techniques in JavaScript


**Using console.log()**

- The simplest way to debug is by printing values to the console.

javascript

```
let num = 10; console.log("The value of num is:", num);
```

- However, console.log() becomes inefficient for large applications


# Using the debugger Keyword


The debugger keyword stops JavaScript execution at a specific point, allowing developers to inspect variables.

let x = 5;let y = 10;debugger;

// Execution pauses here if DevTools is openlet sum = x + y;console.log(sum);The debugger works only if Developer Tools (DevTools) is open in the browser.


# Using the debugger Keyword


## C. Browser Developer Tools (DevTools Debugger)

Most modern browsers like **Google Chrome, Firefox, Edge, and Safari** provide built-in Developer Tools for debugging.

**Steps to Use DevTools Debugger in Chrome:**

1. Open the **Chrome Developer Tools** by pressing `F12` or `Ctrl + Shift + I` (Windows/Linux) or `Cmd + Option + I` (Mac).
2. Go to the "**Sources**" tab.
3. Locate the JavaScript file and add **breakpoints** by clicking on the line number.
4. Refresh the page, and execution will pause at the breakpoint.
5. Use **Step Over (F10), Step Into (F11), and Resume (F8)** to navigate the code execution.


# SNS COLLEGE OF ENGINEERING


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# JavaScript FUNCTIONS

# JAVA SCRIPT FUNCTIONS

- Functions in JavaScript are reusable blocks of code designed to perform specific tasks.
- They allow you to organize, reuse, and modularize code.
- It can take inputs, perform actions, and return outputs.


# Function Syntax and Working


- Below are the rules for creating a function in JavaScript:
- Begin with the keyword **function** followed by,
- A user-defined function name (In the above example, the name is **sum**)
- A list of parameters enclosed within parentheses and separated by commas (In the above example, parameters are **x** and **y**)
- A list of statements composing the body of the function enclosed within curly braces {} (In the above example, the statement is “return x + y”).

# FUNCTIONS

- Parameters are input passed to a function. In the above example, sum() takes two parameters, x and y.
- **Calling Functions**
- After defining a function, the next step is to call them to make use of the function. We can call a function by using the function name separated by the value of parameters enclosed between the parenthesis.




# FUNCTIONS

```
// Function Definition
function welcomeMsg(name) {
    return ("Hello " + name + " welcome to GeeksforGeeks");
}

let nameVal = "User";

// calling the function
console.log(welcomeMsg(nameVal));
```




# WHY FUNCTIONS

**Why Functions?**

- Functions can be used multiple times, reducing redundancy.
- Break down complex problems into manageable pieces.
- Manage complexity by hiding implementation details.
- Can call themselves to solve problems recursively.


# Function Expression



It is similar to a function declaration without the function name. Function expressions can be stored in a variable assignment.

Syntax:

```
let geeksforGeeks= function(paramA, paramB) {
    // Set of statements
}
const mul = function (x, y) {
    return x * y;
};
console.log(mul(4, 5));
```


# ARROW FUNCTIONS


- Arrow functions are a concise syntax for writing functions.

Syntax:

```
let function_name = (argument1, argument2 ,..) => expression
```

```
const a = ["Hydrogen", "Helium", "Lithium", "Beryllium"];
```

```
const a2 = a.map(function (s) {
    return s.length;
});
```




# ARROW FUNCTIONS

```
console.log("Normal way ", a2);

const a3 = a.map((s) => s.length);

console.log("Using Arrow Function ", a3);
```




# Callback Functions

A callback function is passed as an argument to another function and is executed after the completion of that function.

```
function num(n, callback) {
    return callback(n);
}
const double = (n) => n * 2;
console.log(num(5, double));
```


# Anonymous Functions


Anonymous functions are functions without a name. They are often used as arguments to other functions

```
setTimeout(function () {
    console.log("Anonymous function executed!");
}, 1000);
```

# Nested Functions

Functions defined within other functions are called nested functions. They have access to the variables of their parent function.

```
function outerFun(a) {
    function innerFun(b) {
        return a + b;
    }
    return innerFun;
}
const addTen = outerFun(10);
console.log(addTen(5));
```


# SNS COLLEGE OF ENGINEERING


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# Javascript BASICS


# JAVASCRIPT


- JavaScript is a **programming language** used to create dynamic content for websites.
- It is a **lightweight**, **cross-platform**
- **It is a single-threaded** programming language
- JavaScript is an **interpreted** language


# JAVA SCRIPT


- JavaScript on the client side is directly executed in the user's browser.
- Almost all browsers have JavaScript Interpreter and do not need to install any software.
- JavaScript is also used on the Server side (on Web Servers) to access databases, file handling and security features to send responses, to browsers.


# JAVASCRIPT


- The Document Object Model (DOM) provides interfaces for interacting with elements on web pages
- The Browser Object Model (BOM) provides the browser API for interacting with the web browser.
- JavaScript allows you to add interactivity to a web page. Typically, you use JavaScript with HTML and CSS to enhance a web page's functionality, such as validating forms, creating interactive maps, and displaying animated charts.


# JAVASCRIPT


- When a web page is loaded, i.e., after HTML and CSS have been downloaded,
- The JavaScript engine in the web browser executes the JavaScript code.
- The JavaScript code then modifies the HTML and CSS to update the user interface dynamically.
- The JavaScript engine is a component of web browsers responsible for interpreting and executing JavaScript code.


# JAVASCRIPT



- It includes a parser to analyze the code, a compiler to convert it into machine code, and an interpreter to run the compiled code.
- Initially, JavaScript engines were implemented as interpreters.
- However, modern JavaScript engines are commonly implemented as just-in-time compilers that compile JavaScript code to bytecode for improved performance.


# SNS COLLEGE OF ENGINEERING


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## CLIENT vs SERVER JAVA SCRIPT


# Client-side vs. Server-side JavaScript


- When JavaScript is used on a web page, it is executed in web browsers, serving as a client-side language.
- JavaScript can run on both web browsers and servers. A popular JavaScript server-side environment is Node.js. Unlike client-side JavaScript, server-side JavaScript executes on the server and allows you to access databases, file systems, etc.


# SNS COLLEGE OF ENGINEERING


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## JAVASCRIPT BASICS

# BASIC EXAMPLE

Javascript example is easy to code.

JavaScript provides 3 places to put the JavaScript code:

- within body tag,
- within head tag and
- external JavaScript file.


# BASIC EXAMPLE


- The **script** tag specifies that we are using JavaScript.
- The **text/javascript** is the content type that provides information to the browser about the data.
- The **document.write()** function is used to display dynamic content through JavaScript. We will learn about document object in detail later.


# BASIC EXAMPLE


- <script type="text/javascript">
- document.write("JavaScript is a simple language for javatpoint learners");
- </script>


# JavaScript Example : code between the head tag



```
<html>
<head>
<script type="text/javascript">
function msg(){
 alert("Hello Javatpoint");
}
</script>
</head>
```


# JavaScript Example : code between the head tag



```
<html>
<head>
<script type="text/javascript">
function msg(){
 alert("Hello Javatpoint");
}
</script>
</head>
```


# External JavaScript file


- We can create external JavaScript file and embed it in many html page.
- It provides **code re usability** because single JavaScript file can be used in several html pages.
- An external JavaScript file must be saved by .js extension. It increases the speed of the webpage.


# External JavaScript file



- **message.js**
- function msg(){
- alert("Hello Javatpoint");
- }


# External JavaScript file


- <html>
- <head>
- <script type="text/javascript" src="message.js"></script>
- </head>
- <body>
- Welcome to JavaScript
- <form>
- <input type="button" value="click" onclick="msg()"/>
- </form>
- </body>
- </html>


# SNS COLLEGE OF ENGINEERING


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## JAVASCRIPT VARIABLE


# JS VARIABLE


- A **JavaScript variable** is simply a name of storage location.
- The actual value of a variable can be changed at any time.
- Name must start with a letter (a to z or A to Z), underscore( _ ), or dollar( $ ) sign.
- After first letter we can use digits (0 to 9), for example value1.
- JavaScript variables are case sensitive, for example x and X are different variables.

# JS VARIABLE

- **Using Case in Variables:** In JavaScript variables are case-sensitive. **For Example:** "total" and "TOTAL" have different meanings in JavaScript.
- If you are using a variable name that consists of only a word, it is make sure that easier way to use lowercase letters.
- If you are using a variable name with two words such as "total count". It's better to capitalize the first letter of the word. **For example**. "Total_Count", etc.


# JS VARIABLE


- **Allowed special characters:** An important rule to remember is that variable name must start with a letter (a to z or A to Z), underscore( _ ), or dollar( $ ) sign. **For example:** "_totalpay", Total_Count etc.
- After the first letter, we can use digits (0 to 9), **for example:** "value1".
- **Avoiding Reserved Words:** When naming variables in JavaScript avoid the use of the reserved word. **For example:** "if", "case" etc.

# Declaring Variables

- To declare text as a variable, you can use the "var" or "let" keyword. The Following syntax is used for declaring a variable in JavaScript:
- **Syntax:**
- var variable_name;
- In the above syntax, "var" is a keyword and "variable_name" is a name given to a variable.
- **For Example:**
- var total_amount;

# ASSIGNING VALUES

- For assigning a value to a variable, you can use the JavaScript assignment operator (=).
- **Syntax:**
- var variable_name = value;
- In the above syntax, "var" is a keyword, "variable_name" is a name given to a variable and value is used to assign a value to a variable.
- **For Example:**
- var total_amount = 500;
-

# ASSIGNING VALUES



- <script>
- var x = 10;
- var y = 20;
- var z=x+y;
- document.write(z);
- </script>
-

# TYPES OF VARIABLE

- A JavaScript local variable is declared inside block or function. It is accessible within the function or block only. For example:
- <script>
- function abc(){
- var x=10;//local variable
- }
- </script>
- Or,
- <script>
- If(10<13){
- var y=20;//JavaScript local variable
- }
- </script>


# TYPES OF VARIABLE


- JavaScript global variable
- A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable. For example:
- **<**script**>**
- var data=200;//gloabal variable
- function a(){
- document.writeln(data);
- }
- function b(){
- document.writeln(data);
- }
- a();//calling JavaScript function
- b();
- **</**script**>**


# SNS COLLEGE OF ENGINEERING


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## JAVASCRIPT DATATYPES


# JS DATATYPES


- JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.
- Primitive data type
- Non-primitive (reference) data type
- JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine.


# JS DATATYPES


- You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:
- var a=40;//holding number
- var b="Rahul";//holding string

# JavaScript primitive data types

| Data Type | Description |
| --- | --- |
| String | represents sequence of characters e.g. "hello" |
| Number | represents numeric values e.g. 100 |
| Boolean | represents boolean value either false or true |
| Undefined | represents undefined value |
| Null | represents null i.e. no value at all |


# JavaScript nonprimitive data types



| Data Type | Description |
|---|---|
| Object | represents instance through which we can access members |
| Array | represents group of similar values |
| RegExp | represents regular expression |


# ID SELECTOR


- Write a JavaScript function to calculate the sum of two numbers. ...
- Write a JavaScript program to find the maximum number in an array. ...
- Write a JavaScript function to check if a given string is a palindrome (reads the same forwards and backwards).
- Top 50 JavaScript coding Interview Questions and Answers | Keka